ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

APLICACIÓN MÓVIL DE INFORMACIÓN DE FRECUENCIAS DE COOPERATIVAS DE TRANSPORTE INTERPROVINCIAL EVALUADO BAJO PRUEBAS UNITARIAS AUTOMATIZADAS

Trabajo de titulación

Presentado para optar al grado académico de:

INGENIERO EN SISTEMAS INFORMÁTICOS

AUTOR: WILMER HERIBERTO BARRERA QUINCHE

TUTOR: ING. RAÚL ROSERO

Riobamba-Ecuador

2018

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO
FACULTAD DE INFORMÁTICA Y ELECTRÓNICA
ESCUELA DE INGENIERÍA EN SISTEMAS

El tribunal de Tesis certifica que: El trabajo de investigación: APLICACIÓN MÓVIL DE INFORMACIÓN DE FRECUENCIAS DE COOPERATIVAS DE TRANSPORTE INTERPROVINCIAL EVALUADO BAJO PRUEBAS UNITARIAS AUTOMATIZADAS, de responsabilidad del señor Wilmer Heriberto Barrera Quinche, ha sido minuciosamente revisado por los Miembros del Tribunal de Tesis, quedando autorizada su presentación.

	FIRMA	**FECHA**
Ing. Lorena Aguirre **DELEGADA DEL VICEDECANO FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**	_____	_____
Ing. Patricio Moreno **DIRECTOR DE LA ESCUELA DE INGENIERÍA EN SISTEMAS**	_____	_____
Ing. Raúl Rosero **DIRECTOR DEL TRABAJO DE TITULACIÓN**	_____	_____
Dr. Omar Gómez **MIEMBRO DEL TRIBUNAL**	_____	_____

Yo, Wilmer Heriberto Barrera Quinche soy responsable de las ideas, doctrinas y resultados expuestos en este Trabajo de Titulación. Los textos constantes en el documento que provienen de otra fuente están debidamente citados y referenciados.

Como autor, asumo la responsabilidad legal y académica de los contenidos de este trabajo de titulación.

Wilmer Heriberto Barrera Quinche

DEDICATORIA

El presente trabajo de titulación está dedicado principalmente a mis padres Flor Quinche y Heriberto Barrera quienes han sido los pilares fundamentales en mi vida haciendo que sea posible seguir y terminar una carrera universitaria dándome el apoyo necesario día tras día.

Wilmer

AGRADECIMIENTO

El más sincero agradecimiento a mi familia por el apoyo incondicional que me han brindado a lo largo de la carrera como también en cada etapa de mi vida.

Al Ing. Raúl Rosero, al Ing. Omar Gómez por brindarme la asesoría tanto técnica como profesional con su experiencia adquirida a través de los años.

A la ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO, por darme la oportunidad de cumplir una meta personal y profesional en la vida.

Wilmer

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE ANEXOS

RESUMEN

El objetivo del presente trabajo de titulación fue el desarrollo de una aplicación móvil de información de frecuencias de cooperativas de transporte interprovincial evaluado bajo pruebas unitarias automatizadas. Android Studio es el entorno de desarrollo integrado usado en la implementación de esta aplicación móvil, la cual funcionará sobre dispositivos smartphone con sistema operativo Android. El seguimiento del desarrollo se lo ha realizado mediante la metodología ágil scrum, para lo cual se ha diseñado formatos a seguir en el proceso de documentación de la aplicación móvil en cada una de sus etapas. Este sistema se encuentra evaluado tanto con pruebas manuales (pruebas de aceptación definidas en la metodología scrum) como con pruebas unitarias automatizadas, esto con el fin de comparar el tiempo que lleva realizar cada una de estas pruebas sobre un mismo sistema, para esto se ha tomado en cuenta todo el tiempo invertido en pruebas, que a su vez se desglosa en pequeñas fracciones de tiempo denominadas variables, esto con el fin de verificar en cuál de estas existe mayor variación de tiempo entre pruebas manuales y automatizadas; solo funcionalidades del sistema han sido evaluadas bajo estos dos tipos de pruebas, mientras que pruebas de documentación y metáforas se han evaluado con prueba manuales. Junit es el framework utilizado para realizar pruebas automatizadas, este trabaja con el lenguaje de programación Java usado en la creación de aplicaciones móviles Android. Se implementó la automatización de 163 casos de prueba distribuidos en 8 sprints que se llevaron a cabo en un tiempo equivalente al 4% del total de tiempo dedicado a pruebas. Se recomienda hacer un estudio minucioso sobre la geolocalización en dispositivos smartphone con el fin de reducir el consumo de recursos y aumente el grado de precisión en el uso de este servicio.

Palabras clave: <TECNOLOGÍA Y CIENCIAS DE LA INGENIERÍA>, <INGENIERÍA DE SOFTWARE>, <AUTOMATIZACIÓN DE PRUEBAS UNITARIAS>, <APLICACIÓN MÓVIL>, <JUNIT>, <ANDROID>, <METODOLOGÍA DE DESARROLLO ÁGIL SCRUM>

SUMMARY

The objective of the following research work was the development of a mobile application based on information of frequencies in the cooperative interprovincial transport, which is evaluated under automated unitary tests. Android Studio is the integrated development environment used in the implementation of this mobile application, which it will work on smartphone devices with Android operating system. The monitoring of the development has been done through the agile scrum methodology, which it has designed formats to follow in the process of documentation of the mobile application in each of its stages. This system evaluated both manual tests (acceptance tests defined in the scrum methodology) and automated unit tests in order to compare the time it takes to perform each of these tests on the same system, it has been taken into account all the time spent in tests, which in turn is broken down into small fractions of time called variables, that verifies which of these represents the variation of time between manual and automated tests: only functionalities system have been evaluated under these two types of tests, while documentation tests and metaphors have been evaluated with manual tests. Junit is the framework used to perform automated tests, this framework works with Java programming language in the creation of Android mobile applications. We implemented the automation of 163 test cases distributed in 8 sprints that were carried out in an equivalent time of 4% of the total time devoted to tests. It is recommended to make a detailed study on geolocation in smartphone devices in order to reduce the consumption of resources and increase the degree of accuracy in the use of this service.

Keywords: <ENGINEERING TECHNOLOGY AND SCIENCE>, <ENGINEERING SOFTWARE>, <AUTOMATION OF UNIT TESTS>, <MOBILE APPLICATION>, <JUNIT>, <ANDROID>, <METHODOLOGY OF DEVELOPMENT AGILE SCRUM>

INTRODUCCIÓN

En la vida habitual de las personas, el uso de un dispositivo móvil como: Smartphones, Tablet, Smartwatch es imprescindible, ya que, estos proveen un conjunto de herramientas conocidas como aplicaciones, las cuales contribuyen a la solución de problemas cotidianos de las personas, tales como: ver la hora, realizar video llamadas a través de internet, información sobre el clima, incluso como un sistema de posicionamiento global. La evolución en el ámbito de desarrollo móvil Android se ha incrementado exponencialmente, esto debido a que el sistema operativo es gratuito y hasta cierto punto de código abierto, pero con restricciones, esto ha hecho que personas que practican la programación puedan adquirir el conocimiento necesario para desarrollar este tipo de aplicaciones, también gracias a la disponibilidad de información que existe y al costo muy bajo que implica obtener este conocimiento.

Junto con el incremento de desarrollo de aplicaciones móviles surgen métodos o maneras para controlar la fiabilidad y calidad de estos. Por ejemplo, para medir o definir cierto grado de calidad a un producto, al mismo se lo debe evaluar de varias maneras ya sea basándose en normas o reglas ya antes definidas. Este mismo método es por el cual se verifica que un software es o no de calidad, esto con base al cumplimiento o no de los requerimientos establecidos. Al aplicar pruebas al software, estas tienen como objetivo dar un valor a estas, siendo este positivo lo que significa que cumple los requerimientos, así como también negativos significando que dicho software no funciona como se espera que funcione. Las pruebas de software van relacionadas de manera directa con el valor de producto final, así que de cierta manera se puede decir que las pruebas dan valor al producto.

Dentro del desarrollo software en general, gran parte de tiempo y recursos son destinados hacia la fase de pruebas, incluso llegando a usarse más del 50% del total de recursos de un proyecto. Lo que ha llevado que varias empresas de desarrollo no realicen pruebas exhaustivas a su software o también empresas globales optan por liberar versiones beta de un software, para que de esta manera el propio usuario sea quien se encargue de encontrar defectos y errores al software, posteriormente a esto la empresa corregirá dichos fallos y lanzará una versión Alpha con los últimos cambios realizados. Pero la cuestión es, sí aplicar pruebas al software tiene un costo alto en recursos entonces ¿Por qué debemos aplicarlas?, por una simple razón, hacen que el producto sea fiable para los usuarios y al mismo tiempo nos sirven como documentos que respaldan el correcto funcionamiento de un sistema.

Planteamiento del Problema

Antecedentes

Con el desarrollo de sistemas informáticos destinados a solucionar problemas, siendo estos sociales, académicos, de negocios, investigación, u ocio simplemente. Las pruebas o testeo de software han ido evolucionando junto con el desarrollo de estos sistemas, de manera que, al surgir nuevas técnicas y paradigmas de programación para el desarrollo de sistemas informáticos, se ha visto en la necesidad de buscar, nuevos métodos que faciliten la manera de aplicar pruebas al software, pudiendo estar estos en etapas de planificación, desarrollo, o incluso si ya ha sido desarrollado y se encuentre en una etapa de mantenimiento.

Existen organizaciones que apoyan con estándares que definen el proceso de pruebas de software tal como la ISO en la que define lo siguiente. El objetivo de la serie de normas de prueba de software ISO / IEC / IEEE 29119 es definir un conjunto de normas acordadas internacionalmente para las pruebas de software que cualquier organización pueda utilizar cuando realice cualquier tipo de prueba de software (ISO and SC 7, 2013).

También la organización IEEE según el estándar IEEE-829 dice que, Los procesos de prueba determinan si los productos de desarrollo de una actividad dada cumplen con los requisitos de esa actividad y si el sistema y / o el software satisfacen su uso previsto y las necesidades del usuario. Las tareas del proceso de prueba se especifican para diferentes niveles de integridad. Estas tareas de proceso determinan la amplitud y profundidad apropiadas de la documentación de prueba. Los elementos de documentación para cada tipo de documentación de prueba pueden ser seleccionados (IEEE Computer Society et al., 2008, p. 31).

Estos estándares se enfocan en el desarrollo y la documentación de pruebas de software. Mientras en las metodologías convencionales, las pruebas no se ejecutarán hasta que el código esté terminado, aunque puedan diseñarse antes de la codificación, en las ágiles las pruebas se escriben antes comenzar la codificación y sólo se escribe el código necesario para superar las pruebas y guían el proceso de desarrollo. El hecho de que las pruebas se hacen tan pronto como es posible, está alineado con la reducción del impacto del coste de la detección de errores en las fases de pruebas (Yagüe y Garbajosa, 2009, p. 73).

Recursos como: tiempo, costo y personal, son algunos de los factores principales por los cuales, muy pocas veces se realizan pruebas exhaustivas al software, dando como resultado un producto de baja calidad, esto debido al tiempo que toma realizar pruebas manualmente, lo conlleva a elevar

costos en producción o requerir de mayor personal para realizar las mismas. Con el empleo de pruebas automáticas de software, se pretende minimizar este tipo de limitaciones, las cuales se aplican sobre una aplicación móvil desarrollada bajo la plataforma Android.

Formulación del Problema

¿Las pruebas automáticas de software garantizan un producto de calidad?

Sistematización del Problema

¿Las pruebas automáticas de software mejora el tiempo dedicado a este proceso frente a las pruebas manuales tradicionales?
¿Las pruebas unitarias son lo suficientemente flexibles para el manejo de las mismas de forma de agrupada?

Justificación

Justificación Teórica

Para llevar a cabo la aplicación móvil informativa se utilizará las siguientes herramientas tecnológicas como: el IDE de desarrollo Android Studio 2.3.1, Base de datos SQLite 1, MySQL, Framework JUnit 4 destinado a la automatización de pruebas de software. Además de llevar el seguimiento de esta bajo la metodología ágil Scrum, la cual se manejará de manera hibrida en la fase de pruebas, haciendo uso común de las pruebas de aceptación tanto en metáforas del sistema y como en actividades funcionales. En cuanto a la parte extra en la fase de pruebas será solo sobre las actividades funcionales del mismo, las mismas que serán evaluadas de manera automática.

Automatización de pruebas en entornos móviles: Existen diferentes tipos de pruebas que se pueden realizar en aplicaciones que funcionen bajo entornos móviles, siendo estas, pruebas básicas como pruebas de unidad o unitarias, pruebas de regresión, de integración, prueba de interfaz gráfica, etc. En este caso se implementará la automatización de pruebas unitarias de manera automatizada, estas pruebas son las más básicas en las que se evalúa una sola funcionalidad dentro del sistema, para esto se seleccionará todas las funcionalidades dentro del sistema y se obtendrá cada uno de los casos de prueba para evaluar dichas funcionalidades.

Características generales de las herramientas empleadas en el desarrollo de la aplicación móvil:

Android Studio: La edición de códigos de primer nivel, la depuración, las herramientas de rendimiento, un sistema de compilación flexible y un sistema instantáneo de compilación e implementación te permiten concentrarte en la creación de aplicaciones únicas y de alta calidad (Google LLC, 2017a). Además de trabajar bajo la plataforma java, permitiendo así la realización y ejecución de pruebas unitarias e instrumentadas, ya sea ejecutándose en la máquina virtual de java (JVM) o en dispositivos físicos o emulados respectivamente.

Gestor de Base de Datos SQLite: Según el sitio oficial del gestor de base de datos, SQLite es opción popular para el motor de base de datos en teléfonos móviles, PDA, reproductores de MP3, decodificadores y otros aparatos electrónicos. SQLite tiene un tamaño reducido de código, hace un uso eficiente de la memoria, espacio en disco y ancho de banda de disco, es altamente confiable, y no requiere el mantenimiento de un administrador de la base (sqlite.org, 2017).

Framework JUnit: es un framework de prueba para el lenguaje Java. Por sí mismo proporciona la funcionalidad básica para escribir pruebas, que pueden ser mejoradas por otras extensiones. Las pruebas de JUnit también sirven como punto de entrada a casi todos los tipos de pruebas en el desarrollo del proyecto (incluyendo las soluciones auto-desarrolladas antes mencionadas). Es un ciudadano de primera clase con respecto a la forma en que se realizan las pruebas de unidad y el desarrollo impulsado por pruebas. Las pruebas unitarias son una parte de la construcción, y por lo tanto el sistema no puede ser construido (y en consecuencia liberado) cuando algunas de las pruebas no pasan. También hay un efecto secundario positivo, es decir, los desarrolladores se ven obligados a solucionar instantáneamente los problemas detectados por las pruebas unitarias (Marian Jureczko et al., 2016, p. 78).

Metodología ágil SCRUM: La metodología Scrum para el desarrollo ágil de software es un marco de trabajo diseñado para lograr la colaboración eficaz de equipos en proyectos, que emplea un conjunto de reglas y artefactos y define roles que generan la estructura necesaria para su correcto funcionamiento (Cadavid et al., 2013, p. 33).

Justificación Práctica

La aplicación móvil está dirigida hacia las personas que hacen uso habitual del servicio de transporte intercantonal e interprovincial, tratando así de dar una alternativa informativa a través de una aplicación móvil, que sea rápida y fácil de usar. Para cumplir con esto, la aplicación móvil a ser desarrollada se plantea en dos módulos los cuales son: *Módulo de aplicación móvil informativa de Cooperativas de transporte público* y *Módulo de pruebas automáticas de software*

Dentro de las actividades que corresponden al *Módulo de aplicación móvil informativa de Cooperativas de transporte público*

- Búsqueda de horario de servicio de transporte basado en una cooperativa de transporte específica.
- Reporte de lista de cooperativas de transporte
- Reporte de lista de rutas que cubre un transporte
- Reporte de lista de horarios de una ruta específica
- Búsqueda de horario de servicio de transporte basado en una ciudad destino
- Reporte de lista de ciudades origen de una ruta
- Reporte de lista de ciudades destino de una ruta
- Reporte de próximo horario de salida de servicio de transporte en una ruta específica
- Obtener mi localización actual
- Obtener la ciudad más cercana a mi localización
- Sincronizar app móvil con base de datos, ubicado en el servicio de hosting
- Reporte detalles de una ruta específica y horario específico
- Agregar ruta a rutas favoritas
- Contactar a agencia de una cooperativa de transporte
- Verificar la disponibilidad de boletos online de una cooperativa de transporte, con base en la ruta y horario
- Reporte de lista de rutas favoritas
- Reporte de detalle de una ruta favorita

Módulo de pruebas automáticas de software son:

- Análisis de casos de pruebas sobre la app móvil
- Diseño de casos de pruebas sobre app móvil
- Implementación de pruebas sobre app móvil, usando framework JUnit 4, junto con el IDE Android Studio

El sistema propuesto tiene como línea de investigación de la EIS (Escuela de Ingeniería en Sistemas), Proceso de desarrollo de Software basado en el ámbito de Construcción del Software. También se ajusta a la línea de investigación de la ESPOCH (Escuela Superior Politécnica de Chimborazo) en las Tecnologías de la información, comunicación. Y se adapta al Plan Nacional del Buen Vivir (PNBV) teniendo como objetivo Mejorar la calidad de vida de la población y su política 3.12.e. Propiciar la ampliación de la oferta del transporte público masivo e integrado, en sus diferentes alternativas, para garantizar el acceso equitativo de la población al servicio.

Objetivos

Objetivo General

Desarrollar una aplicación móvil de información de frecuencias de cooperativas de transporte interprovincial evaluado bajo pruebas unitarias automatizadas.

Objetivos Específicos

- Determinar pruebas unitarias a emplear en aplicación móvil
- Diseñar pruebas unitarias que serán aplicadas
- Implementar pruebas unitarias automatizadas sobre aplicación móvil
- Desarrollar aplicación móvil bajo metodología ágil scrum

CAPÍTULO I

1 MARCO TEÓRICO REFERENCIAL

1.1 Sistemas Operativos Móviles

1.1.1 Sistema operativo Android

Android es el sistema operativo de la compañía Google.Inc. Este sistema operativo es el mismo que funciona sobre celulares smartphone, smartwatch, autos y otros dispositivos que se usa a en nuestro día a día. A parte de ser un sistema operativo gratuito, muchas cooperativas celulares han optado por usar este sistema en sus productos.

Este sistema operativo se torna realmente atractivo por diversas características, entre ellas se encuentran:

- Plataforma totalmente libre basado en Linux que permite desarrollar aplicaciones y/o modificar las ya existentes con lenguaje de Java.
- Es multitasking permitiendo mantener distintas aplicaciones corriendo al mismo tiempo.
- Compatible con una gran variedad de hardware en el mercado (tablets y dispositivos celulares de marcas como: Motorola, Samsung, ZTE, Huawei, Ericsson por nombrar algunas) permitiendo al usuario elegir el dispositivo que mejor se ajusta a sus necesidades.
- Posee un portal llamado Play Store donde se tiene acceso a muchas aplicaciones que pueden ser utilizadas.
- Permite realizar actualizaciones del sistema operativo en línea siempre y cuando el dispositivo soporte los requerimientos de este.
- Puede operar soluciones tecnológicas referentes al uso de redes sociales, mensajería instantánea, correo electrónico, modificación y lectura de procesadores de palabras, hojas de cálculo, presentaciones, lectura de formatos pdf, entre otros.
- Se puede conseguir mucha información a través de documentos web o libros.
- Como característica importante, cuenta con el gran apoyo y la capacidad tecnológica proporcionada por su principal socio “Google” (Malave Polanco y Beauperthuy Taibo, 2011, p. 82).

1.1.2 Sistema operativo IOS (Iphone OS)

El iPhone OS es un sistema operativo propietario. Apple no licencia iPhone OS a otros fabricantes de dispositivos, solo su empresa OEM obtiene autorización para usar iPhone OS. Apple usa su sistema operativo para obtener control sobre su producto, y ve el iPhone y el sistema operativo iPhone como un paquete en la competencia de teléfonos inteligentes, por lo que la venta de iPhone es la mayor preocupación de Apple, la prioridad de iPhone OS viene en segundo lugar (Lin y Ye, 2009, p. 619).

1.2 Bases de datos

1.2.1 Gestor de base de datos MySQL

MySQL es la base de datos de código abierto más popular del mundo. Con su rendimiento comprobado, fiabilidad y facilidad de uso, MySQL se ha convertido en la principal opción de base de datos para aplicaciones basadas en web, utilizada por propiedades web de alto perfil, como Facebook, Twitter, YouTube y los cinco principales cinco sitios web. Además, es una opción extremadamente popular como base de datos integrada (Oracle Corporation, 2018a).

- **MySQL es un sistema de gestión de base de datos.**

Una base de datos es una colección estructurada de datos. Puede ser cualquier cosa, desde una simple lista de compras hasta una galería de imágenes o la gran cantidad de información en una red corporativa. Para agregar, acceder y procesar datos almacenados en una base de datos informática, necesita un sistema de administración de bases de datos como el servidor MySQL. Dado que las computadoras son muy buenas para manejar grandes cantidades de datos, los sistemas de administración de bases de datos juegan un papel central en la informática, como utilidades independientes o como parte de otras aplicaciones (Oracle Corporation, 2018b).

- **Las bases de datos MySQL son relacionales.**

Una base de datos relacional almacena datos en tablas separadas en lugar de poner todos los datos en un gran almacén. Las estructuras de la base de datos están organizadas en archivos físicos optimizados para la velocidad. El modelo lógico, con objetos como bases de datos, tablas, vistas, filas y columnas, ofrece un entorno de programación flexible. Establece reglas que rigen las relaciones entre diferentes campos de datos, como uno a uno, uno a muchos, único, requerido u opcional, y "punteros " entre tablas diferentes. La base de datos impone estas reglas, de modo que, con una base de datos bien diseñada, su aplicación nunca vea datos inconsistentes, duplicados, huérfanos, desactualizados o faltantes (Oracle Corporation, 2018b).

- **El software MySQL es de código abierto.**

Código abierto significa que es posible que cualquier persona use y modifique el software. Cualquiera puede descargar el software MySQL de Internet y usarlo sin pagar nada. Si lo desea, puede estudiar el código fuente y modificarlo para adaptarlo a sus necesidades. El software MySQL usa la GPL (Licencia pública general de GNU), para definir lo que puede y no puede hacer con el software en diferentes situaciones. Si no se siente cómodo con la GPL o necesita insertar el código MySQL en una aplicación comercial, puede comprar una versión con licencia comercial de nuestra parte (Oracle Corporation, 2018b).

- **El servidor de base de datos MySQL es muy rápido, confiable, escalable y fácil de usar.**

El servidor MySQL puede ejecutarse cómodamente en una computadora de escritorio o portátil, junto con otras aplicaciones, servidores web, etc., que requieren poca o ninguna atención. Si dedica una máquina completa a MySQL, puede ajustar la configuración para aprovechar toda la memoria, la potencia de la CPU y la capacidad de E/S disponibles. MySQL también puede escalar hasta clusters de máquinas, conectadas en red (Oracle Corporation, 2018b).

- **El servidor MySQL funciona en sistemas cliente/servidor o integrados.**

El software de base de datos MySQL es un sistema cliente/servidor que consiste en un servidor SQL multiproceso que admite diferentes back-ends, varios programas de cliente y bibliotecas, herramientas administrativas y una amplia gama de interfaces de programación de aplicaciones. También proporcionamos el servidor MySQL como una biblioteca incrustada de subprocesos múltiples que puede vincular a su aplicación para obtener un producto autónomo más pequeño, más rápido y más fácil de administrar (Oracle Corporation, 2018b).

Dentro de las principales características revisadas y que sobresalen del gestor de base de datos relacional MySql son:

Internos y portabilidad

- Escrito en C y C ++.
- Probado con una amplia gama de compiladores diferentes.
- Para la portabilidad, utiliza **CMake** en MySQL 5.5 y superior. Las series anteriores usan GNU Automake, Autoconf y Libtool.
- Utiliza un diseño de servidor de varias capas con módulos independientes.
- Diseñado para ser completamente multiproceso usando hilos de kernel, para usar fácilmente múltiples CPU si están disponibles.

- Proporciona motores de almacenamiento transaccionales y no transaccionales.
- Utiliza tablas de disco B-tree muy rápidas (MyISAM) con compresión de índice.
- Diseñado para que sea relativamente fácil agregar otros motores de almacenamiento. Esto es útil si desea proporcionar una interfaz SQL para una base de datos interna.
- Utiliza un sistema de asignación de memoria basado en subprocesos muy rápido.
- Ejecuta uniones muy rápidas utilizando una combinación optimizada de bucle anidado.
- Implementa tablas hash en memoria, que se usan como tablas temporales.
- Implementa funciones SQL usando una biblioteca de clases altamente optimizada que debe ser lo más rápido posible. Por lo general, no hay asignación de memoria después de la inicialización de la consulta.
- Proporciona el servidor como un programa separado para su uso en un entorno de red cliente/servidor, y como una biblioteca que puede integrarse (vincularse) en aplicaciones independientes. Tales aplicaciones se pueden usar de forma aislada o en entornos donde no hay una red disponible (Oracle Corporation, 2018c).

Seguridad

- Un sistema de privilegios y contraseñas que es muy flexible y seguro, y que permite la verificación basada en el host.
- Seguridad de contraseña mediante el cifrado de todo el tráfico de contraseñas cuando se conecta a un servidor (Oracle Corporation, 2018c).

Escalabilidad y límites

- Soporte para grandes bases de datos. Utiliza el servidor MySQL con bases de datos que contienen 50 millones de registros. También sabemos de los usuarios que usan MySQL Server con 200,000 tablas y alrededor de 5,000,000,000 de filas.
- Soporte para hasta 64 índices por mesa. Cada índice puede constar de 1 a 16 columnas o partes de columnas. El ancho máximo del índice para InnoDBtablas es 767 bytes o 3072 bytes (Oracle Corporation, 2018c).

Conectividad

- Los clientes pueden conectarse al servidor MySQL usando varios protocolos:
- Los clientes pueden conectarse utilizando sockets TCP/IP en cualquier plataforma.

- En los sistemas Windows, los clientes pueden conectarse utilizando conductos con nombre si el servidor se inicia con la --enable-named-pipe opción. Los servidores Windows también admiten conexiones de memoria compartida si se inician con la --shared-memory opción. Los clientes pueden conectarse a través de la memoria compartida mediante el uso de la --protocol=memory opción.
- En los sistemas Unix, los clientes pueden conectarse utilizando archivos de socket de dominio Unix.
- Los programas cliente de MySQL se pueden escribir en muchos idiomas. Una biblioteca de cliente escrita en C está disponible para clientes escritos en C o C ++, o para cualquier lenguaje que proporcione enlaces de C.
- Las API para C, C ++, Eiffel, Java, Perl, PHP, Python, Ruby y Tcl están disponibles, permitiendo que los clientes de MySQL se escriban en varios idiomas.
- La interfaz Connector/ODBC (MyODBC) proporciona compatibilidad con MySQL para los programas cliente que usan conexiones ODBC (Conectividad de base de datos abierta). Por ejemplo, puede usar MS Access para conectarse a su servidor MySQL. Los clientes se pueden ejecutar en Windows o Unix. La fuente Connector/ODBC está disponible. Todas las funciones de ODBC 2.5 son compatibles, al igual que muchos otros.
- La interfaz Connector/J proporciona soporte MySQL para programas de cliente Java que usan conexiones JDBC. Los clientes se pueden ejecutar en Windows o Unix. La fuente Connector/J está disponible.
- MySQL Connector/Net permite a los desarrolladores crear fácilmente aplicaciones .NET que requieren conectividad de datos segura y de alto rendimiento con MySQL. Implementa las interfaces ADO.NET requeridas y las integra en herramientas con ADO.NET. Los desarrolladores pueden crear aplicaciones usando su elección de lenguajes .NET. MySQL Connector/Net es un controlador ADO.NET totalmente administrado, escrito en C # 100% puro (Oracle Corporation, 2018c).

Localización

- El servidor puede proporcionar mensajes de error a los clientes en muchos idiomas.
- Soporte completo para distintos conjuntos de caracteres, incluyendo latin1(CP1252), German, big5, ujis, varios conjuntos de caracteres Unicode, y mucho más. Por ejemplo, los caracteres escandinavos " å", " ä" y "ö" están permitidos en los nombres de tabla y columna.
- Todos los datos se guardan en el juego de caracteres elegido.

- La ordenación y las comparaciones se realizan de acuerdo con el conjunto de caracteres predeterminado y la intercalación. es posible cambiar esto cuando se inicia el servidor MySQL (Oracle Corporation, 2018c).

Clientes y herramientas

MySQL incluye varios programas de cliente y utilidad. Estos incluyen tanto programas de línea de comandos como **mysqldump, mysqladmin**, y programas gráficos como MYSQL Workbench.

- El servidor MySQL tiene soporte integrado para declaraciones de SQL para verificar, optimizar y reparar tablas. Estas declaraciones están disponibles desde la línea de comandos a través del cliente mysqlcheck. MySQL también incluye myisamchk, una herramienta de línea de comandos muy rápida para realizar estas operaciones en MyISAM tablas (Oracle Corporation, 2018c).

1.2.2 Gestor de base de datos SQLite

SQLite es una biblioteca en proceso que implementa un motor de base de datos SQL transaccional independiente, sin servidor y de configuración cero. El código para SQLite es de dominio público y, por lo tanto, es gratuito para cualquier uso, comercial o privado. SQLite es la base de datos más implementada del mundo con más aplicaciones de las que podemos contar, incluidos varios proyectos de alto perfil (Sqlite.org, 2017).

Entre las principales características del gestor de base de datos SQLite están:

- SQLite es un motor de base de datos SQL incorporado.
- SQLite no tiene un proceso de servidor por separado.
- SQLite lee y escribe directamente en archivos de disco ordinarios.
- Una base de datos SQL completa con múltiples tablas, índices, disparadores y vistas, está contenida en un solo archivo de disco.
- El formato de archivo de la base de datos es multiplataforma: puede copiar libremente una base de datos entre sistemas de 32 bits y de 64 bits.
- El código fuente es totalmente gratuito para cualquiera que lo desee, pero también cuenta con soporte profesional (Sqlite.org, 2017).

SQLite es una biblioteca compacta. Con todas las funciones habilitadas, el tamaño de la biblioteca puede ser inferior a 500 Kb, según la plataforma de destino y la configuración de optimización del compilador. Existe una compensación entre el uso de la memoria y la velocidad. SQLite

generalmente se ejecuta más rápido cuanta más memoria le dé. Sin embargo, el rendimiento suele ser bastante bueno incluso en entornos con poca memoria. Dependiendo de cómo se use, SQLite puede ser más rápido que la E/S directa del sistema de archivos (Sqlite.org, 2017).

1.3 Patrón de arquitectura Modelo, Vista, Presentador (MVP)

El patrón MVP es un tipo de patrón de diseño de UI que permite el desacoplamiento de la lógica de UI sin representación de UI y permite la base de código de tratamiento sin tener en cuenta la tecnología de presentación utilizada. Una de sus ventajas es el hecho de que desacoplando el código en el nivel del presentador estamos permitiendo la reutilización de la lógica del código a tecnologías de presentación totalmente diferentes (Pau et al., 2010, p. 173).

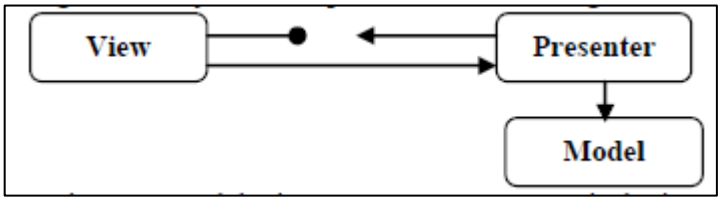


Figura 1-1: Modelo Vista Presentador.

Fuente: (Pau et al., 2010, p. 173).

- La vista contiene la instancia de Presentador (ver "sabe presentador");
- Presentador es la única clase que sabe cómo llegar para modelar y recuperar los datos necesarios para llevar a cabo la lógica comercial;
- Presentador se comunica con la Vista a través de la interfaz de visualización (representación abstracta de la vista sin atributos específicos de UI) (Pau et al., 2010, p. 173);
- La vista no conoce el modelo. Debido a esto, hay un bajo acoplamiento entre el Modelo y la Vista. Significa que, si se cambió el Modelo o la Vista, no es necesario modificar otra parte, siempre y cuando las interfaces sean estables. Esto también representa la flexibilidad de la arquitectura y la reutilización de la lógica de negocios en el modelo (Zhang y Luo, 2010, p. 532).

Con otras palabras, La vista es responsable de la representación visual del formulario y contiene un presentador en campo privado. La vista está implementando la interfaz cuyos miembros son aproximados y la tecnología muestra elementos de la interfaz de usuario de la vista (Pau et al., 2010, p. 173). La vista es simulable para fines de prueba. En la tradición, es imposible probar el componente Vista o lógica de negocio antes de que otro se haya completado debido al estrecho acoplamiento entre Vista y Lógica de negocio. Por la misma razón, la unidad de prueba para la vista o el componente de lógica de negocios es difícil. Todos esos problemas son resueltos por el

patrón MVP. En MVP, no hay dependencia directa entre la Vista y el Modelo. Por esa razón, el desarrollador podría usar el objeto simulado para inyectar en Ver o Modelo para que puedan ser probados por uno mismo (Zhang y Luo, 2010, p. 532).

Arquitectura

Como un patrón, MVP tiene varias formas expresivas y arquitecturas cuando se implementa en diferentes plataformas, el implemento de concreto está restringido por las características de la plataforma, y la consistencia es otro factor que debe considerarse al diseñar el modelo de arquitectura de concreto (Zhang y Luo, 2010, p. 532).

Este modelo está compuesto por cinco partes:

IView: es la abstracción de View que se compone de un conjunto de reglas que declaran qué datos y funciones se deben implementar en View. En general, cada componente de vista tiene su propio componente IView (Zhang y Luo, 2010, p. 532).

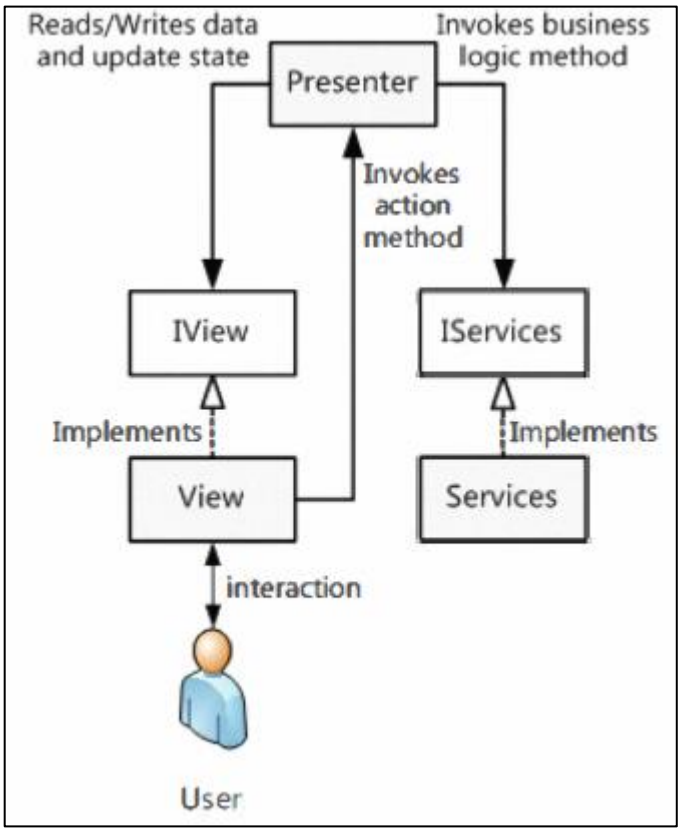


Figura 2-1: Arquitectura de Modelo Vista Presentador.
Fuente: (Zhang y Luo, 2010, p. 533).

View es la parte que interactúa con los usuarios. Se pueden usar muchas tecnologías para implementar View. Cada componente de View debe implementar el IView homólogo y un componente de IView podría tener muchos implementos con diferentes tecnologías de VI, como resultado, las Vistas implementadas desde el mismo IView podrían tomar el lugar del otro (Zhang y Luo, 2010, p. 533).

IServices es un conjunto de interfaces que definen las funciones que deben implementarse mediante componentes de lógica de negocios. Se corresponde con la interfaz del Modelo (Zhang y Luo, 2010, p. 533).

Los **services** son los componentes de lógica de negocio que implementa IServices. View y Presenter necesitan la ayuda de los Servicios para realizar negocios de aplicaciones debido a que no tienen ninguna lógica de negocio. Los servicios que no tienen nada que ver con la tecnología VI comúnmente, el instrumento concreto de ellos son algunas clases generales. Algunas veces los Servicios necesitan un Repositorio para manejar la operación de la base de datos, esto está fuera del rango de este documento (Zhang y Luo, 2010, p. 533).

Presentador es el núcleo del patrón de MVP. Al igual que los Servicios, Presenter es ajeno a la tecnología VI porque solo depende de IView. El presentador recibe las acciones del usuario y lee los datos de entrada desde la vista, y luego invoca funciones en Servicios para completar la lógica comercial y modifica el estado de la Vista. Todo este trabajo se llama lógica de presentación (Zhang y Luo, 2010, p. 533).

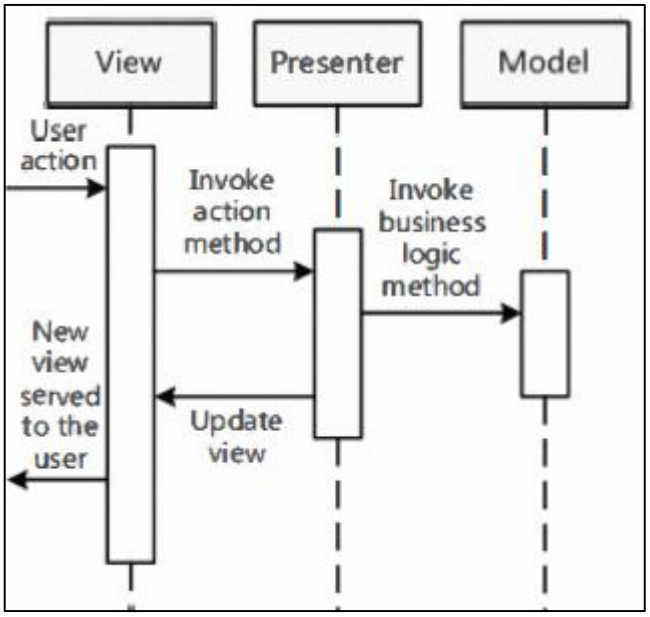


Figura 3-1: Diagrama de secuencia del patrón MVP.

Fuente: (Zhang y Luo, 2010, p. 533).

1.4 Modelo de pruebas

1.4.1 Pruebas de software

El probar o testeo de software es una de las partes más importantes dentro de cualquier sistema informática, ya que de ello depende que un producto final tengo calidad aceptable o no. Las pruebas no pueden probar la ausencia de fallas. Para hacer esto, una prueba necesitaría ejecutar un programa en cada situación posible con cada valor de entrada posible y con todas las condiciones posibles. En la práctica, una prueba completa o exhaustiva no es factible. Debido a los efectos combinatorios, el resultado de esto es un número casi infinito de pruebas. Tal "prueba" para todas las combinaciones no es posible. Por lo tanto, en la práctica, no es posible probar incluso un pequeño programa de forma exhaustiva. Solo es posible considerar una parte de todos los casos de prueba imaginables. Pero, aun así, las pruebas todavía representan una gran parte del esfuerzo de desarrollo. Sin embargo, una generalización de la extensión del esfuerzo de prueba es difícil porque depende mucho del carácter del proyecto (Spillner et al., 2014, p. 14).

Las pruebas de software se basan en ciertos principios los cuales debemos tomar en cuenta a la hora de llevar a cabo testeo de software sobre un sistema. A continuación, se describen estos:

Las pruebas muestran la presencia de defectos, no su ausencia: Las pruebas pueden mostrar que el producto falla, es decir, que hay defectos. Las pruebas no pueden probar que un programa esté libre de defectos. Las pruebas adecuadas reducen la probabilidad de que defectos ocultos estén presentes en el objeto de prueba. Incluso si no se encuentran fallas durante las pruebas, esto no es prueba de que no haya defectos (Spillner et al., 2014, p. 33).

Exhaustiva prueba es imposible: Es imposible ejecutar una prueba exhaustiva que incluya todos los valores posibles para todas las entradas y sus combinaciones combinadas con todas las condiciones previas diferentes. El software, en la práctica normal, requeriría un número "astronómicamente" alto de casos de prueba. Cada prueba es solo una muestra. Por lo tanto, el esfuerzo de prueba debe controlarse, teniendo en cuenta el riesgo y las prioridades (Spillner et al., 2014, p. 34).

Las actividades de prueba deben comenzar lo más temprano posible: Las actividades de prueba deben comenzar lo más temprano posible en el ciclo de vida del software y enfocarse en los objetivos definidos. Esto contribuye a encontrar defectos temprano (Spillner et al., 2014, p. 34).

Agrupación de defectos: Los defectos no están distribuidos uniformemente; se agrupan juntos. La mayoría de los defectos se encuentran en algunas partes del objeto de prueba. Por lo tanto, si se detectan muchos defectos en un lugar, normalmente hay más defectos cerca. Durante la prueba, uno debe reaccionar de manera flexible a este principio (Spillner et al., 2014, p. 34).

La paradoja del pesticida: Los insectos y las bacterias se vuelven resistentes a los pesticidas. De manera similar, si las mismas pruebas se repiten una y otra vez, tienden a perder su efectividad: no descubren nuevos defectos. Los defectos antiguos o nuevos pueden estar en partes del programa no ejecutadas por los casos de prueba. Para mantener la efectividad de las pruebas y para combatir esta "paradoja de los pesticidas", se deben desarrollar y agregar a la prueba casos de prueba nuevos y modificados. Las partes del software que aún no se han probado o las combinaciones de entradas que no se usaron anteriormente se verán afectadas y se podrán encontrar más defectos (Spillner et al., 2014, p. 34).

La prueba depende del contexto: Las pruebas deben adaptarse a los riesgos inherentes al uso y al entorno de la aplicación. Por lo tanto, no se deben probar dos sistemas exactamente de la misma manera. La intensidad de las pruebas, los criterios de salida de las pruebas, etc. deben decidirse individualmente para cada sistema de software, dependiendo de su entorno de uso. Por ejemplo, los sistemas críticos para la seguridad requieren diferentes pruebas que las aplicaciones de comercio electrónico (Spillner et al., 2014, p. 35).

No fallas significa que el sistema es útil es una falacia: Encontrar fallas y reparar defectos no garantiza que el sistema cumpla con las expectativas y necesidades del usuario. La participación temprana de los usuarios en el proceso de desarrollo y el uso de prototipos son medidas preventivas destinadas a evitar este problema (Spillner et al., 2014, p. 35).

Asignación de recursos en pruebas de software

En los recursos asignados a proyectos de software se puede decir que en las fases de pruebas esta lleva un alto consumo de recursos tanto en recursos humano, tiempo y costos, incluso a veces hasta llegar a un costo mayor del 50% del total del proyecto. A continuación, se expone ejemplos de gasto de recursos en el área de testeo:

- Para algunos proyectos importantes con más de 10 años-persona de esfuerzo, la codificación y la prueba en conjunto usaron el 40%, y un 8% adicional se utilizó para la integración. En proyectos intensivos en pruebas (por ejemplo, sistemas críticos para la seguridad), el esfuerzo de prueba aumentó hasta un 80% del presupuesto total.

- En un proyecto, el esfuerzo de prueba fue 1.2 veces más alto que el esfuerzo de codificación, con dos tercios del esfuerzo de prueba utilizado para la prueba de componentes.
- Para otro proyecto en la misma empresa de desarrollo de software, el costo de prueba del sistema fue del 51,9% del proyecto (Spillner et al., 2014, p. 15).

El esfuerzo de prueba a menudo se muestra como la proporción entre el número de probadores y el número de desarrolladores. La proporción varía de 1 probador por 10 desarrolladores a hasta 3 probadores por desarrollador. La conclusión es que los esfuerzos de prueba o el presupuesto gastado para las pruebas varían enormemente. El esfuerzo de prueba puede crecer muy grande. Los gerentes de pruebas se enfrentan al dilema de posibles casos de prueba y variantes de casos de prueba convirtiéndose rápidamente en cientos o miles de pruebas. Este problema también se llama explosión combinatoria o explosión de caso de prueba. Además de la restricción necesaria en el número de casos de prueba, el gerente de prueba normalmente tiene que luchar con otro problema: la falta de recursos (Spillner et al., 2014, pp. 15–17).

Los sistemas con altos riesgos deben probarse más a fondo que los sistemas que no generan grandes pérdidas si fallan. La evaluación de riesgos debe hacerse para las partes individuales del sistema, o incluso para posibilidades de error único. Si existe un alto riesgo de fallas por un sistema o subsistema, debe haber un mayor esfuerzo de prueba que para los (sub) sistemas menos críticos. Las normas internacionales para la producción de sistemas críticos para la seguridad utilizan este enfoque para exigir que se apliquen diferentes técnicas de prueba para software de diferentes niveles de integridad. Para un productor de un juego de computadora, guardar puntajes de juego erróneos puede significar un riesgo muy alto, incluso si no se realiza un daño real, porque los clientes no confiarán en un juego defectuoso. Esto conduce a grandes pérdidas de ventas, tal vez incluso para todos los juegos producidos por la compañía. Por lo tanto, para cada programa de software se debe decidir qué tan intensa y minuciosamente se debe probar. Esta decisión debe tomarse en función del riesgo esperado de falla del programa. Como no es posible realizar una prueba completa, es importante cómo se usan los recursos de prueba limitados. Para obtener un resultado satisfactorio, las pruebas deben diseñarse y ejecutarse de forma estructurada y sistemática. Solo entonces es posible encontrar muchas fallas con el esfuerzo adecuado y evitar pruebas innecesarias que no brinden más información sobre la calidad del sistema (Spillner et al., 2014, p. 16).

Planificación y control de pruebas

La ejecución de una tarea tan importante como las pruebas no debe tener lugar sin un plan. La planificación del proceso de prueba comienza al comienzo del proyecto de desarrollo de software.

Al igual que con toda planificación, durante el transcurso del proyecto, los planes previos deben revisarse, actualizarse y ajustarse periódicamente. La misión y los objetivos de las pruebas deben definirse y acordarse, así como los recursos necesarios para el proceso de prueba. Se debe organizar o ajustar una estructura organizacional con la administración de pruebas apropiada si es necesario. El control de la prueba es el control de las actividades de prueba y la comparación de lo que sucede realmente durante el proyecto con el plan. Incluye informar el estado de las desviaciones del plan y tomar las medidas necesarias para alcanzar los objetivos planificados en la nueva situación. El plan de prueba debe actualizarse a la situación cambiada (Spillner et al., 2014, pp. 19–20).

Parte de las tareas de administración de pruebas es administrar y mantener el proceso de prueba, la infraestructura de prueba y el software de prueba. El seguimiento del progreso puede basarse en los informes adecuados de los empleados, así como en los datos generados automáticamente a partir de las herramientas. Los acuerdos sobre estos temas deben hacerse temprano. La tarea principal de la planificación es determinar la estrategia o el enfoque de la prueba. Como no es posible realizar una prueba exhaustiva, las prioridades deben establecerse en función de la evaluación del riesgo. Las actividades de prueba se deben distribuir a los subsistemas individuales, según el riesgo esperado y la gravedad de los efectos de falla. Los subsistemas críticos deben recibir una mayor atención, por lo tanto, deben probarse más intensamente. Para subsistemas menos críticos, pruebas menos extensas pueden ser suficientes. Si no se esperan efectos negativos en caso de falla, las pruebas podrían omitirse en algunas partes. Sin embargo, esta decisión debe tomarse con gran cuidado. El objetivo de la estrategia de prueba es la distribución óptima de las pruebas a las partes "correctas" del sistema de software (Spillner et al., 2014, p. 20).

Análisis y diseño de prueba

La primera tarea es revisar la base de la prueba, es decir, la especificación de lo que se debe probar. La especificación debe ser lo suficientemente clara y concreta para desarrollar casos de prueba. La base para la creación de una prueba puede ser la especificación o los documentos de arquitectura, los resultados del análisis de riesgos u otros documentos producidos durante el proceso de desarrollo del software. Por ejemplo, un requisito puede ser demasiado impreciso para definir el resultado esperado o el comportamiento esperado del sistema. No se pueden desarrollar casos de prueba. La capacidad de prueba de este requisito es insuficiente. Por lo tanto, debe ser revisado. La determinación de las condiciones previas y los requisitos para el diseño del caso de prueba debe basarse en un análisis de los requisitos, el comportamiento esperado y la estructura del objeto de prueba (Spillner et al., 2014, p. 22).

Al igual que con el análisis de la base para una prueba, el objeto de prueba también debe cumplir ciertos requisitos para ser simple de probar. La capacidad de prueba debe ser revisada. Este proceso incluye verificar la facilidad con la que se pueden abordar las interfaces y la facilidad con la que el objeto de prueba se puede separar en unidades más pequeñas y fáciles de probar. Estos problemas deben abordarse durante el desarrollo y el objeto de prueba debe diseñarse y programarse en consecuencia. Los resultados de este análisis también se utilizan para establecer y priorizar las condiciones de la prueba en función de los objetivos generales de la prueba. Las condiciones de prueba establecen exactamente qué se probará. Esta puede ser una función, un componente o alguna característica de calidad (Spillner et al., 2014, p. 22).

La estrategia de prueba determinada en el plan de prueba define qué técnicas de prueba se utilizarán. La estrategia de prueba depende de los requisitos de confiabilidad y seguridad. Si existe un alto riesgo de falla para el software, se deben planificar pruebas muy exhaustivas. Si el software es menos crítico, prueba puede ser menos formal. En la especificación de la prueba, los casos de prueba se desarrollan utilizando las técnicas de prueba especificadas. Se utilizan las técnicas planificadas previamente, así como las técnicas elegidas en función de un análisis de la posible complejidad en el objeto de prueba. Es importante garantizar la trazabilidad entre las especificaciones que se probarán y las pruebas en sí. Debe quedar claro qué casos de prueba evalúan qué requisitos y viceversa. Solo de esta manera es posible decidir qué requisitos deben ser o han sido probados, con qué intensidad y con qué casos de prueba. Incluso se debe verificar la trazabilidad de los cambios de requisitos en los casos de prueba y viceversa. La especificación de los casos de prueba se lleva a cabo en dos pasos. Los casos de prueba lógica deben definirse primero. Después de eso, los casos de prueba lógica pueden traducirse en casos concretos de pruebas físicas, lo que significa que se seleccionan las entradas reales (casos de prueba concretos). Además, la secuencia opuesta es posible: desde los físicos hasta los casos de prueba lógica general. Este procedimiento se debe utilizar si un objeto de prueba se especifica de manera insuficiente y la especificación de prueba se debe realizar de una manera bastante experimental. El desarrollo de casos de pruebas físicas, sin embargo, es parte de la siguiente fase, la implementación de pruebas (Spillner et al., 2014, pp. 22–23).

La base de prueba guía la selección de casos de prueba lógica con todas las técnicas de prueba. Los casos de prueba se pueden determinar a partir de la especificación del objeto de prueba (técnicas de diseño de prueba de caja negra) o se pueden crear analizando el código fuente (técnicas de diseño de prueba de caja blanca). Resulta claro que la actividad llamada especificación de casos de prueba puede tener lugar en momentos totalmente diferentes durante el proceso de desarrollo del software. Esto depende de las técnicas de prueba elegidas, que se encuentran en la estrategia de prueba. Las tareas de planificación, análisis y diseño de pruebas

pueden y deben tener lugar en paralelo con las actividades de desarrollo anteriores. Para cada caso de prueba, debe describirse la situación inicial (precondición). Debe quedar claro qué condiciones ambientales deben cumplirse para la prueba. Además, antes de la ejecución de la prueba, se debe definir qué resultados y comportamientos se esperan. Los resultados incluyen salidas, cambios en los datos y estados globales (persistentes) y cualquier otra consecuencia del caso de prueba. Para definir los resultados esperados, el probador debe obtener la información de alguna fuente adecuada. En este contexto, esto a menudo se llama un oráculo u oráculo de prueba. Un oráculo de prueba es un mecanismo para predecir los resultados esperados. La especificación puede servir como un oráculo de prueba (Spillner et al., 2014, p. 23). Hay dos posibilidades principales:

- El probador deriva los datos esperados en base a la especificación del objeto de prueba.
- Si las funciones que realizan la acción inversa están disponibles, se pueden ejecutar después de la prueba y luego el resultado se verifica con la entrada original. Un ejemplo de este escenario es el cifrado y descifrado de datos (Spillner et al., 2014, p. 23).

Los casos de prueba se pueden diferenciar por dos criterios:

- Primero son los casos de prueba para examinar el comportamiento, el resultado y la reacción especificados. Aquí se incluyen casos de prueba que examinan el manejo específico de excepciones y casos de error (prueba negativa). Pero a menudo es difícil crear las condiciones previas necesarias para la ejecución de estos casos de prueba.
- Luego están los casos de prueba para examinar la reacción de los objetos de prueba a entradas o condiciones no válidas e inesperadas, que no tienen un manejo de excepciones especificado (Spillner et al., 2014, p. 24).

Prevención versus detección

La calidad no se puede lograr mediante la evaluación de un producto ya completado. El objetivo, por lo tanto, es evitar defectos o deficiencias de calidad en primer lugar, y hacer que los productos sean evaluables mediante medidas de control de calidad. Algunas medidas de control de calidad incluyen: estructurar el proceso de desarrollo con un estándar de desarrollo de software y apoyar el proceso de desarrollo con métodos, técnicas y herramientas. Los errores no detectados en el software que causaron millones de pérdidas en los negocios han requerido el crecimiento de pruebas independientes, que realiza una empresa que no sean los desarrolladores del sistema. Además de las evaluaciones de productos, las evaluaciones de procesos son esenciales para un programa de gestión de calidad. Los ejemplos incluyen documentación de estándares de codificación, prescripción y uso de estándares, métodos y herramientas, procedimientos para

respaldo de datos, metodología de prueba, gestión de cambios, documentación de defectos y reconciliación (Lewis y Veerapillai, 2005, p. 6).

La gestión de calidad disminuye los costos de producción porque cuanto antes se localice y corrija un defecto, menos costoso será a largo plazo. Con la llegada de las herramientas de prueba automatizadas, aunque la inversión inicial puede ser sustancial, el resultado a largo plazo será un producto de mayor calidad y costos de mantenimiento reducidos. El costo total de una gestión de calidad efectiva es la suma de cuatro costos de componentes: prevención, inspección, falla interna y falla externa. Los costos de prevención consisten, en primer lugar, en acciones para evitar defectos. Los costos de inspección consisten en medir, evaluar y auditar productos o servicios para cumplir con los estándares y especificaciones. Los costos de falla interna son aquellos incurridos en la reparación de productos defectuosos antes de que sean entregados. Los costos de falla externa consisten en los costos de los defectos descubiertos después de que el producto ha sido liberado. El último puede ser devastadores porque pueden dañar la reputación de la organización o resultar en la pérdida de ventas futuras. La mayor recompensa es con la prevención. Aumentar el énfasis en los costos de prevención reduce la cantidad de defectos que no se detectan al cliente, mejora la calidad del producto y reduce el costo de producción y mantenimiento (Lewis y Veerapillai, 2005, pp. 6–7).

Verificación versus Validación

La verificación está demostrando que un producto cumple con los requisitos especificados durante actividades previas llevadas a cabo correctamente durante el ciclo de vida de desarrollo, y la validación comprueba que el sistema cumpla con los requisitos del cliente al final del ciclo de vida. Es una prueba de que el producto cumple con las expectativas de los usuarios y garantiza que el sistema ejecutable funcione según lo especificado. La creación del producto de prueba está mucho más relacionada con la validación que con la verificación. Tradicionalmente, las pruebas de software se han considerado un proceso de validación, es decir, una fase del ciclo de vida. Una vez completada la programación, el sistema se valida o prueba para determinar su funcionamiento funcional y operativo. Cuando la verificación se incorpora a las pruebas, las pruebas se realizan durante todo el ciclo de vida de desarrollo. Para obtener los mejores resultados, es una buena práctica combinar la verificación con la validación en el proceso de prueba. La verificación incluye procedimientos sistemáticos de revisión, análisis y pruebas, empleados durante todo el ciclo de vida del desarrollo del software, comenzando con la fase de requisitos del software y continuando hasta la fase de codificación. La verificación garantiza la calidad de la producción y el mantenimiento del software. Además, la verificación impone una práctica de desarrollo organizada y sistemática de tal manera que el programa resultante puede ser fácilmente entendido y evaluado por una parte independiente (Lewis y Veerapillai, 2005, p. 7).

La verificación surgió hace unos 20 años como resultado de la necesidad de la industria aeroespacial de software extremadamente confiable en sistemas en los que un error en un programa podría causar fallas en la misión y provocar enormes reveses financieros y de tiempo, o incluso situaciones que amenazan la vida. El concepto de verificación incluye dos criterios fundamentales: el software debe realizar adecuada y correctamente todas las funciones previstas, y el software no debe realizar ninguna función que, por sí sola o en combinación con otras funciones, pueda degradar el rendimiento de todo el sistema. El objetivo general de la verificación es garantizar que cada producto de software desarrollado a lo largo del ciclo de vida del software cumpla con las necesidades y objetivos del cliente, tal como se especifica en el documento de requisitos del software. La verificación también establece la capacidad de tratamiento entre las diversas secciones de la documentación del software y las partes asociadas de la especificación de requisitos. Un esfuerzo de verificación exhaustivo garantiza que todos los requisitos de calidad y rendimiento del software en la especificación se prueban adecuadamente y que los resultados de la prueba se pueden repetir después de que se hayan instalado los cambios. La verificación es un "proceso de mejora continua" y no tiene una terminación definitiva. Se debe usar durante todo el ciclo de vida del sistema para mantener la configuración y la integridad operativa (Lewis y Veerapillai, 2005, pp. 8–9).

La verificación garantiza que el software funciona según lo previsto y tiene los atributos necesarios (por ejemplo, portabilidad) y aumenta las posibilidades de que el software contenga pocos errores (es decir, un número aceptable en el producto final). Proporciona un método para supervisar de cerca el proyecto de desarrollo de software y proporciona a la administración un estado detallado del proyecto en cualquier momento. Cuando se utilizan procedimientos de verificación, la administración puede estar segura de que los desarrolladores siguen un proceso de desarrollo de software formal, secuencial y rastreable, con un conjunto mínimo de actividades para mejorar la calidad del sistema. Una crítica de la verificación es que aumenta considerablemente los costos de desarrollo de software. Sin embargo, cuando se considera el costo del software a lo largo de todo el ciclo de vida desde el inicio hasta el abandono final del sistema, la verificación en realidad reduce el costo total del software. Con un programa de verificación efectivo, normalmente hay una reducción de cuatro a uno en los defectos del sistema instalado. Debido a que las correcciones de errores pueden costar de 20 a 100 veces más durante las operaciones y el mantenimiento que durante el diseño, los ahorros generales superan con creces el gasto adicional inicial (Lewis y Veerapillai, 2005, p. 9).

1.4.1.1 Desarrollo guiado por pruebas (DGP/TDD)

¿Qué es TDD?

TDD es la práctica de escribir sus pruebas antes de escribir cualquier código de aplicación. TDD es una práctica que los defensores y teóricos de software ágiles defienden con mayor y más frecuencia. Es uno de los principales pilares de la metodología ágil. Se han realizado varios estudios a lo largo de los años y se han publicado muchos libros blancos, lo que indica claramente que el uso de TDD ha resultado en un código de aplicación más sólido y exitoso (Sood, 2016, p. 7). El desarrollo guiado por pruebas (TDD) tiene un conjunto de características por las cuales tiene una amplia aceptación a la hora de ser implementado siguiendo una metodología ágil. Uno de sus objetivos es tener un código limpio que funcione. Conduce a una mejor calidad y menos defectos en el código. Elimina la necesidad de pasar días en un depurador para buscar errores difíciles de encontrar. Por lo tanto, reduce los esfuerzos de depuración (Paranj, 2017, p. 12).

Practicar TDD significa diseñar software de manera que pueda ser probado en cualquier momento bajo automatización. El diseño para la capacidad de prueba en TDD es una llamada más alta que el diseño de un código "bueno" porque el código comprobable es un buen código. Las estrategias tradicionales de prueba rara vez afectan el diseño del código de producción, son onerosas para los desarrolladores y probadores, y a menudo dejan las pruebas al final de un proyecto donde las limitaciones presupuestarias y de tiempo amenazan las pruebas exhaustivas. Test Driven Development invierte sistemáticamente estos patrones (Karlesky et al., 2006, p. 1).

Como TDD se sigue con el fin de automatizar pruebas unitarias en cada una de las funcionalidades o características del sistema. Con cada característica del sistema abordada, el código de prueba de la unidad se agrega a un conjunto de pruebas automatizadas. Las pruebas de regresión completas pueden tener lugar todo el tiempo. Las pruebas unitarias automatizadas atrapan a la mayoría de los errores a un nivel bajo y dejan a la mente humana problemas difíciles de prueba como colisiones de tiempo o interacciones inesperadas del subsistema (Karlesky et al., 2006, p. 2).

¿Por qué TDD?

La necesidad de TDD surge del hecho de que puede haber cambios constantes en el código de la aplicación. Esto se convierte en un problema mayor cuando uso el proceso de desarrollo ágil, ya que es inherentemente un proceso de desarrollo iterativo (Sood, 2016, p. 8).

Pasos para aplicar TDD

Kent Beck es el creador de la programación extrema, una metodología de desarrollo de software que evita las especificaciones formales rígidas para un proceso de diseño colaborativo e iterativo.

Ken Beck resume los cinco pasos de TDD de la siguiente manera:

- Agregue rápidamente una prueba
- Ejecute todas las pruebas y vea que la nueva falla. Como todavía no hay un código para que la prueba pase, esta prueba fallará.
- Haga un pequeño cambio para pasar la prueba lo más rápido posible.
- Ejecute todas las pruebas y vea que todas tienen éxito
- Refactorizar para eliminar la duplicación (Paranj, 2017, p. 13).

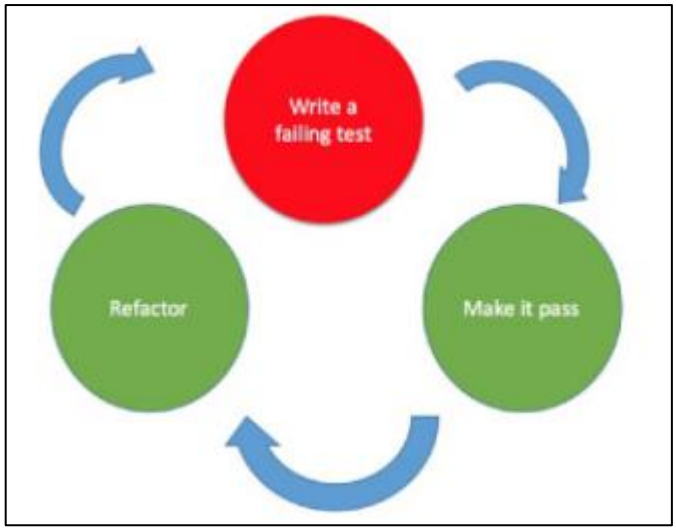


Figura 4-1: Pasos repetitivos que forman el ciclo TDD.

Fuente: (Paranj, 2017, p. 13).

En el primer paso TDD, escribimos una prueba. En el segundo paso, registramos un requisito como prueba. También diseñamos explícitamente la API del cliente. Diseñar la API aquí significa responder preguntas tales como las siguientes:

- ¿El nombre del método revela la intención?
- ¿Debería ser un método de instancia o un método de clase?
- ¿Cuáles son los parámetros de este método?
- ¿Cuáles son los parámetros requeridos?
- ¿Cuáles son los parámetros opcionales?
- ¿Debería pasar este parámetro al constructor en lugar de pasarlo al método?
- ¿Debería el parámetro tener un valor predeterminado? (Paranj, 2017, p. 13).

Las respuestas a estas preguntas se convierten en tus decisiones de diseño que expresas en el código. En el tercer paso. Siempre escribimos el código posible simple que hace pasar la prueba. Esto nos permite mantener nuestras opciones abiertas y evolucionar nuestro diseño. En el cuarto

paso, el foco está en cumplir los requisitos. En el quinto paso, la atención se centra en crear un buen diseño (Paranj, 2017, p. 14).

Prueba fallida

¿Por qué comenzar con una prueba fallida? La prueba que sigue a las citas de Kent Beck resume la razón de escribir la prueba antes de escribir el código de producción: Escribir una prueba fallida es una forma de probar la prueba. Si todas las pruebas están aprobadas, nos informa que no hay problemas conocidos con nuestro código. Al escribir una prueba para exponer una deficiencia, estamos aclarando el problema. También estamos esbozando el diseño. Cuando probamos primero, reducimos el estrés, lo que nos hace más probables de evaluar (Paranj, 2017, p. 14).

Escribir una prueba fallida

Escribir una prueba de falla son los pasos uno y dos de los cinco pasos de TDD, hágase las siguientes preguntas:

- ¿Cuál es la responsabilidad de nuestro sistema bajo prueba (SBP)?
- ¿Qué debería hacer?
- ¿Cuál es la API para hacer que el SBP haga esto?
- ¿Qué necesita el SBP para cumplir con su responsabilidad?
- ¿Qué salida hay para observar?
- ¿Cómo podemos decir que funcionó correctamente?
- ¿Cómo se define la corrección? (Paranj, 2017, p. 14).

TDD ofrece varios beneficios claros:

- El código siempre se prueba.
- El sistema crece orgánicamente a medida que se gana más conocimiento del sistema.
- Los desarrolladores pueden agregar nuevas características o modificar el código existente con la confianza de que las pruebas de regresión automatizadas revelarán fallas y resultados inesperados (Karlesky et al., 2006, p. 2).

- **Calidad del código:** la prueba en tdd hace que el programador confíe más en su código. Los programadores pueden estar seguros de la corrección sintáctica y semántica de su código. Las pruebas controlan el diseño del código (Sood, 2016, p. 9). Como efecto secundario, el código tiende a mejorarse debido al desacoplamiento necesario para crear código comprobable (Karlesky et al., 2006, p. 2).

- **Arquitectura en evolución:** un código de aplicación puramente basado en pruebas da paso a una arquitectura en evolución. Esto significa que no tenemos que predefinir nuestros límites arquitectónicos y los patrones de diseño. A medida que la aplicación crece, también lo hace la arquitectura. Esto da como resultado una aplicación flexible para futuros cambios (Sood, 2016, p. 9).

- **Documentar el código:** estas pruebas también documentan los requisitos y el código de la aplicación. Los puristas ágiles normalmente consideran los comentarios dentro del código como un "olor". Según ellos, tus pruebas deberían documentar tu código (Sood, 2016, p. 9). El conocimiento del sistema se captura en las pruebas; las pruebas son documentación "viviente" (Karlesky et al., 2006, p. 2).

- **Una anulación de la ingeniería:** prueba escrita antes de que el código de la aplicación defina y documente los límites. Dado que todos los límites están predefinidos en las pruebas, es difícil escribir código de aplicación que incumpla estos límites. Esto, sin embargo, asume que TDD se está siguiendo religiosamente (Sood, 2016, p. 9).

- **Cambio de paradigma:** cuando comencé con TDD, noté que la primera pregunta que me hice después de ver los problemas fue "¿cómo puedo resolverla?". Sin embargo, esto es contraproducente, TDD obliga al programador a pensar en la capacidad de prueba de la solución antes de su implementación. Esto daría como resultado una solución, que es comprobable por naturaleza, ya que hemos utilizado pruebas para derivar estas soluciones. Comprender cómo probar un problema significaría una mejor comprensión del problema y sus casos límite. Esto puede dar como resultado el refinamiento de los requisitos o el descubrimiento de algunos requisitos nuevos. Ahora me resulta imposible no pensar en la capacidad de prueba de los problemas antes de la solución. Ahora la primera pregunta que me hago es; "¿Cómo puedo probarlo?" Esto se puede hacer traduciendo los requisitos en pruebas (Sood, 2016, p. 9).

- **Código de mantenimiento:** siempre me ha resultado más fácil trabajar en una aplicación que históricamente ha sido impulsada por pruebas en lugar de una que no lo ha hecho. ¿Por qué? Solo porque cuando realizo cambios en el código existente, las pruebas existentes me aseguro de no romper funcionalmente ninguno. Esto da como resultado un código altamente mantenible, donde muchos programadores pueden colaborar simultáneamente (Sood, 2016, p. 9).

- **Refactorizar libremente:** tener una buena cobertura de prueba sobre el código de la aplicación permite al programador refactorizar y mejorar continuamente el código, manteniendo al mismo tiempo una naturaleza aplicando la misma prueba varias veces y obteniendo el mismo resultado del código de la aplicación (Sood, 2016, p. 9).

1.4.1.2 Aplicación de pruebas en aplicaciones móviles

El término prueba móvil se refiere a diferentes tipos de pruebas, como pruebas de aplicaciones móviles nativas, pruebas de dispositivos móviles y pruebas de aplicaciones web móviles. Usamos pruebas de aplicaciones móviles para referirnos a "actividades de prueba para aplicaciones nativas y web en dispositivos móviles usando herramientas y métodos de prueba de software bien definidos para garantizar la calidad en funciones, comportamientos, rendimiento y calidad de servicio, así como características, tales como movilidad, usabilidad, interoperabilidad, conectividad, seguridad y privacidad" (Gao et al., 2014, p. 46).

Para hacer frente a las actualizaciones frecuentes de dispositivos y tecnologías móviles, los ingenieros necesitan reutilizar y un entorno rentable para probar aplicaciones móviles y una infraestructura elástica para admitir la automatización de pruebas a gran escala (Gao et al., 2014, p. 54).

La mayoría de las investigaciones sobre pruebas de aplicaciones móviles se han centrado en soluciones a problemas técnicos específicos en las siguientes áreas:

- Caja blanca y pruebas unitarias de aplicaciones móviles.
- Caja negra y prueba GUI de aplicaciones móviles.
- Validación de los requisitos de calidad de servicio (QoS) de la aplicación móvil.
- Prueba de red inalámbrica.
- Prueba de usabilidad móvil.
- Marcos de automatización de pruebas móviles (Gao et al., 2014, p. 46).

Requisitos

Varios requisitos únicos distinguen las pruebas de aplicaciones móviles de las pruebas de software convencionales. Primero, las aplicaciones móviles deben funcionar correctamente en cualquier momento y en cualquier lugar. Además, debido a que los servicios móviles a menudo se desarrollan para un conjunto de dispositivos seleccionados, las aplicaciones deben funcionar correctamente en todas las plataformas que tienen, por ejemplo, diferentes sistemas operativos, tamaños de pantallas, recursos de energía computacional y duración de la batería. Además, para

brindar la rica experiencia que los usuarios de dispositivos móviles esperan, las aplicaciones móviles deben incluir múltiples canales de entrada (teclado, voz y gestos), soporte multimedia y otras características de usabilidad mejoradas. Además, se requiere simulación y virtualización a gran escala para mantener bajos los costos de hardware. Finalmente, debido a que la mayoría de los planes de servicios móviles admiten una variedad de redes inalámbricas (2G, 3G, 4G, Wi-Fi, WiMax), las aplicaciones móviles deben funcionar en diversos contextos de conectividad de red (Gao et al., 2014, pp. 46–48).

Pruebas de actividades y objetivos

Teniendo en cuenta estos requisitos, las pruebas de aplicaciones móviles tienden a centrarse en las siguientes actividades y objetivos:

Pruebas de funcionalidad y comportamiento: actividades que validan funciones de servicio, API web móviles, comportamientos externos del sistema, inteligencia basada en el sistema e interfaces de usuario (UI) (Gao et al., 2014, p. 48).

Pruebas de QoS: actividades que evalúan la carga, el rendimiento, la confiabilidad / disponibilidad, la escalabilidad y el rendimiento del sistema. Pruebas de interoperabilidad: actividades que verifican la interoperabilidad del sistema entre diferentes dispositivos, plataformas, navegadores y redes inalámbricas (Gao et al., 2014, p. 48).

Pruebas de usabilidad e internacionalización: actividades que evalúan el contenido y las alertas de la IU, los flujos y escenarios de operación del usuario, la riqueza de los medios y el soporte de interacción de gestos (Gao et al., 2014, p. 48).

Pruebas de seguridad y privacidad: actividades que verifican la autenticación del usuario, la seguridad del dispositivo, la seguridad de la sesión, la penetración del sistema / red, la seguridad de las comunicaciones móviles punto a punto (P2P), la seguridad de las transacciones de extremo a extremo y la privacidad del usuario (Gao et al., 2014, p. 48).

Pruebas de movilidad: actividades que validan funciones basadas en la ubicación, perfiles de usuario, datos del sistema y datos del usuario (Gao et al., 2014, p. 48).

Pruebas de compatibilidad y conectividad: actividades que evalúan la compatibilidad del navegador móvil y la plataforma y la conectividad de red inalámbrica diversa.

Prueba de multitenencia: actividades que validan las funciones basadas en el inquilino, comportamientos del sistema, datos del sistema y UI (Gao et al., 2014, p. 48).

Pruebas nativas versus aplicación web

Dos tipos de software de aplicaciones móviles pueden someterse a pruebas. Las aplicaciones nativas se implementan y ejecutan en dispositivos móviles y generalmente dependen de las API y los dongles nativos, como la cámara y las API de GPS. Las aplicaciones web consisten en un servidor de aplicaciones y un software cliente ejecutado sobre navegadores web, a través del cual los usuarios pueden acceder a los servicios de las aplicaciones. Las pruebas de los dos tipos deben tener en cuenta sus diferencias (Gao et al., 2014, p. 50).

Objetivos de prueba primarios

Las pruebas nativas de aplicaciones tienen como objetivo validar la calidad de las aplicaciones móviles descargadas y ejecutadas en plataformas móviles seleccionadas en diferentes dispositivos móviles. Las pruebas se centran en la funcionalidad y el comportamiento (incluidas las funciones específicas del dispositivo, como la interacción gestual), los requisitos de QoS, la usabilidad, la seguridad y la privacidad (Gao et al., 2014, p. 50).

Funciones nativas para dispositivos móviles

Dado que las aplicaciones dependen de plataformas y dispositivos móviles nativos, pueden desarrollarse para admitir ciertas funciones basadas en API nativas, como transacciones con tarjetas de crédito y validación de huellas dactilares. Estos son enfoques específicos en las pruebas nativas de aplicaciones móviles; por lo general, no se prueban en aplicaciones web móviles porque son tan específicas del dispositivo (Gao et al., 2014, p. 51).

Las pruebas de aplicaciones web tienen como objetivo validar la calidad de las aplicaciones web móviles con diferentes navegadores web en diversos dispositivos móviles. A diferencia de las aplicaciones nativas, las aplicaciones web suelen proporcionar a los usuarios un cliente móvil delgado para acceder a las funciones en línea desde el servidor de fondo. Por lo tanto, además de la funcionalidad y el comportamiento, los requisitos de QoS, la usabilidad, la seguridad y la privacidad, las pruebas de aplicaciones web móviles se centran en la conectividad y la interoperabilidad (Gao et al., 2014, p. 51).

Entorno de prueba

Para las aplicaciones nativas de los dispositivos móviles, la plataforma móvil subyacente o el sistema operativo constituyen el entorno de prueba. Para lograr una automatización efectiva, las soluciones de prueba deben ser compatibles, implementables y ejecutables en múltiples

plataformas. Para las aplicaciones web móviles, el navegador web subyacente es el entorno de prueba (Gao et al., 2014, p. 50).

Interfaces de usuario

La prueba de UI para aplicaciones móviles nativas considera clientes móviles gordos o inteligentes, contenido multimedia y gráficos, y funciones de gestos. Para las aplicaciones web móviles, las pruebas de interfaz de usuario consideran clientes móviles delgados basados en la web, clientes móviles descargables y compatibilidad con gráficos y medios enriquecidos basados en navegador (Gao et al., 2014, p. 51).

Tecnología para la automatización de pruebas móviles

Las tecnologías que permiten la prueba automatizada de aplicaciones móviles nativas incluyen lenguajes de programación como Java (Android), Objective-C (iOS) y Visual C ++ (Windows Mobile); SDK estandarizados y herramientas de desarrollo de proveedores de dispositivos; y plataformas de desarrollo de aplicaciones. Para las aplicaciones web móviles, las tecnologías de automatización de pruebas generalmente se encuentran en línea en forma de HTML5, CSS3 y JavaScript. El desarrollador de la aplicación también puede usar lenguajes o marcos del lado del servidor como PHP, Rails y Python (Gao et al., 2014, p. 51).

Entre las principales pruebas que se realizan a aplicaciones móviles tenemos:

Prueba de conectividad móvil: Diversas redes inalámbricas móviles admiten las necesidades de conectividad de aplicaciones nativas y web. Debido a que la conectividad afecta el rendimiento de las aplicaciones y la interoperabilidad, los ingenieros deben prestarle atención durante las pruebas. Para las aplicaciones móviles nativas, esto significa considerar la sincronización de contenido en línea durante las pruebas, así como los problemas de descarga, implementación y personalización relacionados con las tiendas de aplicaciones y los mercados. Para las aplicaciones web móviles, es importante probar teniendo en cuenta la conectividad a Internet y diversos protocolos inalámbricos, así como las conexiones móviles entre clientes y servidores (Gao et al., 2014, p. 51).

Prueba de usabilidad: La validación de una aplicación nativa móvil generalmente implica probar gestos basados en dispositivos móviles, contenido, interfaces y la experiencia general del usuario. Por el contrario, las pruebas de usabilidad para aplicaciones web móviles generalmente se centran en el contenido de la GUI basada en la web, las interfaces y los flujos de operación del usuario (Gao et al., 2014, p. 51).

Pruebas de movilidad: Las pruebas de movilidad en un dispositivo nativo generalmente implican probar las funciones, características, datos, perfiles y API basados en la ubicación del dispositivo (Gao et al., 2014, p. 51).

Problemas, desafíos y necesidades

A pesar de su creciente importancia, el campo de las pruebas de aplicaciones móviles sigue enfrentando numerosos problemas, desafíos y necesidades.

Entornos de prueba: Según los comentarios de los ingenieros de prueba, la construcción de entornos de prueba móviles aún implica altos costos y niveles de complejidad. Configurar un entorno de prueba móvil para múltiples aplicaciones en cada plataforma móvil para una variedad de dispositivos es tedioso, lento y costoso, y las actualizaciones frecuentes tanto en el dispositivo como en los espacios de la plataforma solo agravan este desafío. Esto plantea una necesidad clave en el campo de pruebas de aplicaciones móviles: un enfoque reutilizable y sistemático para probar la configuración y configuración del entorno que cubre una amplia gama de dispositivos y plataformas móviles. Esta solución podría eliminar las tediosas operaciones manuales actuales y reducir los costos también (Gao et al., 2014, p. 54).

El ambiente reutilizable ideal tendría las siguientes características:

- Conectividad unificada a diferentes plataformas móviles en dispositivos móviles
- Una capacidad de configuración que admite la implementación, instalación y ejecución sistemáticas de una aplicación determinada en diferentes plataformas para dispositivos móviles específicos;
- Diversas opciones de configuración de red móvil (Gao et al., 2014, p. 54).

Las pruebas de la aplicación web móvil requieren validación con diferentes navegadores y tecnologías web, lo que significa que los ingenieros de prueba también deben evaluar diferentes parámetros de QoS. Este tipo de prueba requiere una forma de generar y simular solicitudes de aplicaciones móviles a gran escala e interacciones con infraestructuras y soluciones de prueba reales o virtuales. En consecuencia, los ingenieros deben poder seleccionar y configurar diversas opciones de conectividad de red inalámbrica de la manera más transparente posible (Gao et al., 2014, p. 54).

Automatización de prueba

Las pruebas automatizadas de aplicaciones móviles plantean dos problemas adicionales: la falta de estandarización en la infraestructura de prueba móvil, los lenguajes de scripting y los

protocolos de conectividad entre las herramientas y plataformas de prueba móviles; y la falta de una infraestructura de automatización de pruebas unificadas y soluciones que atraviesen plataformas y navegadores en la mayoría de los dispositivos móviles (Gao et al., 2014, p. 54).

Esto también se remonta al desafío del entorno de prueba: para hacer frente a las actualizaciones frecuentes de dispositivos y tecnologías móviles, los ingenieros necesitan un entorno reutilizable y rentable para realizar pruebas en dispositivos móviles, así como una infraestructura elástica para admitir pruebas a gran escala automatización. En última instancia, esto requerirá nubes de prueba móviles basadas en dispositivos equipadas con dispositivos móviles conectados, diversos y reemplazables; nubes de emulación escalables que pueden permitir la creación, implementación y control de pruebas de emulación móvil a gran escala; y una solución de control y ejecución de prueba unificada que admite la automatización de pruebas simultáneas y a gran escala (Gao et al., 2014, p. 54).

1.4.1.3 Pruebas en IDE de desarrollo Android Studio

Android Studio está diseñada para simplificar las pruebas. Con solo algunos clics, puedes establecer una prueba JUnit que se ejecute en el JVM local o una prueba instrumentada que se ejecute en un dispositivo. Por supuesto, también puedes extender tus capacidades de pruebas integrando frameworks de prueba como Mockito, para probar llamadas de Android API en tus pruebas de unidades locales, y Espresso o UI Automator para ejercitar la interacción con usuarios en tus pruebas instrumentadas. Puedes generar pruebas Espresso automáticamente usando la grabadora de pruebas Espresso (Google LLC, 2018a).

La compilación de Gradle interpreta estos conjuntos de orígenes de pruebas como lo hace en el caso de los conjuntos de orígenes de app de tu proyecto, que te permiten crear pruebas según las variantes de compilación. Cuando creas un proyecto nuevo o agregas un módulo de app, Android Studio crea los conjuntos de orígenes de pruebas e incluye un ejemplo de archivo de prueba en cada uno (Google LLC, 2018a).

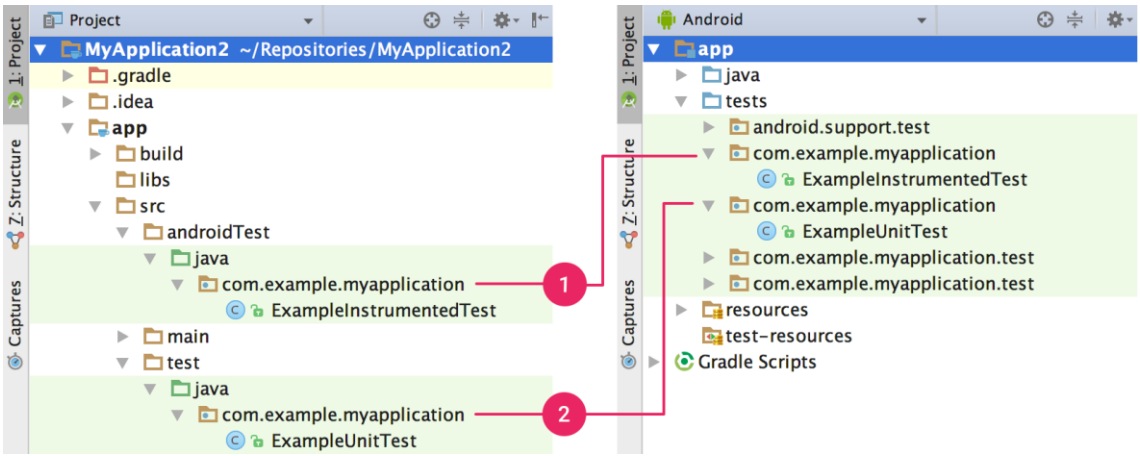


Figura 5-1: Pruebas instrumentadas (1) del proyecto y (2) pruebas JVM locales.

Fuente: (Google LLC, 2018a).

Pirámide de pruebas

Esta pirámide ilustra cómo su aplicación debe incluir las tres categorías de pruebas: pequeña, mediana y grande:

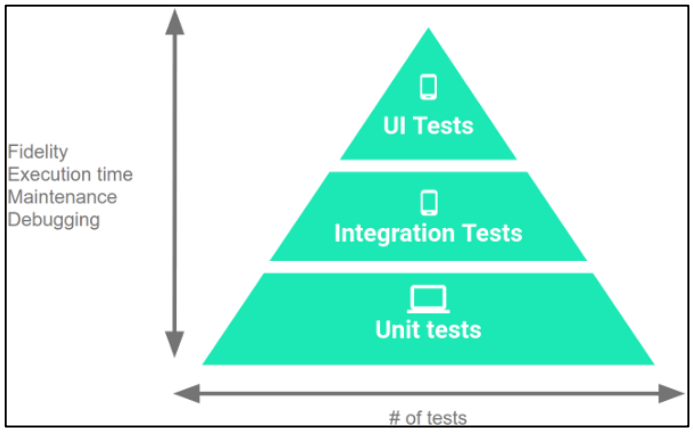


Figura 6-1: Pirámide de pruebas.

Fuente: (Google LLC, 2018b).

- Las pruebas pequeñas son pruebas unitarias que puede ejecutar de manera aislada de los sistemas de producción. Por lo general, se burlan de todos los componentes principales y deben ejecutarse rápidamente en su máquina.
- Las pruebas medianas son pruebas de integración que se encuentran entre las pruebas pequeñas y las pruebas grandes. Integran varios componentes y funcionan con emuladores o dispositivos reales.
- Las pruebas grandes son pruebas de integración y UI que se ejecutan al completar un flujo de trabajo de IU. Aseguran que las tareas clave del usuario final funcionen como se esperaba en emuladores o dispositivos reales (Google LLC, 2018b).

Aunque las pruebas pequeñas son rápidas y están enfocadas, lo que le permite solucionar las fallas rápidamente, también son de baja fidelidad y autónomas, lo que dificulta tener la confianza de que una prueba aprobada le permite a su aplicación funcionar. Encuentra el conjunto opuesto de concesiones al escribir pruebas grandes (Google LLC, 2018b).

Debido a las diferentes características de cada categoría de prueba, debe incluir pruebas de cada capa de la pirámide de prueba. Aunque la proporción de pruebas para cada categoría puede variar en función de los casos de uso de su aplicación, generalmente recomendamos la siguiente división entre las categorías: 70 por ciento pequeño, 20 por ciento mediano y 10 por ciento grande (Google LLC, 2018b).

1.4.1.4 Pruebas de unidad local

Teniendo en cuenta que el presente proyecto se desarrollará sobre una plataforma Android, el presente trabajo se presenta las maneras y formas en las que se puede trabajar con pruebas automáticas dentro de esta plataforma, esto a través del IDE de desarrollo Android Studio. Dentro de las pruebas que se pueden llevar a cabo para Android están las pruebas de unidad local, estas se encuentran definidas en la propia documentación del IDE de desarrollo Android Studio.

La ubicación del directorio dentro del Proyecto Android es el siguiente:
Ubicación: module-name/src/test/java/.

Estas son pruebas que se ejecutan en la máquina virtual Java (JVM) local de tu máquina. Usa estas pruebas para minimizar el tiempo de ejecución cuando tus pruebas no tengan dependencias de marco de Android o cuando puedas simular las dependencias del marco de Android (Google LLC, 2018a).

En el tiempo de ejecución, estas pruebas se ejecutan en una versión modificada de android.jar en la que se quitan todos los modificadores del tipo final. Esto te permite usar bibliotecas de simulación populares, como Mockito (Google LLC, 2018a).

1.4.1.5 Pruebas instrumentadas

Otra de las pruebas dentro de Android Studio son las pruebas instrumentadas, que se verá a continuación.

La ubicación del directorio dentro del Proyecto Android es el siguiente:

Ubicación: module-name/src/androidTest/java/.

Estas son pruebas que se ejecutan en un dispositivo o emulador de hardware. Estas pruebas tienen acceso a las API Instrumentation, y te permiten acceder a información como el Context de la app que pruebas y controlar la app a prueba desde tu código de prueba. Usa estas pruebas cuando escribas pruebas de IU funcionales e integradoras para automatizar la interacción de usuarios, o cuando tus pruebas tengan dependencias de Android que los objetos ficticios que no puedan contemplar (Google LLC, 2018a).

Debido a que las pruebas instrumentadas se compilan en un APK (por separado del APK de tu app), deben tener su propio archivo AndroidManifest.xml. Sin embargo, Gradle automáticamente genera este archivo durante la compilación para que no se vea en el conjunto de fuentes de tu proyecto. Puedes agregar tu propio archivo de manifiesto si es necesario, por ejemplo, a fin de especificar un valor diferente para `minSdkVersion` o registrar receptores de ejecución solo para tus pruebas. Cuando se compila tu app, Gradle combina varios archivos de manifiesto en un manifiesto (Google LLC, 2018a).

También puede ejecutar pruebas de unidades instrumentadas en un dispositivo físico o emulador, lo que no implica ninguna burla o punteo del marco. Debido a que esta forma de prueba implica tiempos de ejecución significativamente más lentos que las pruebas de unidades locales, sin embargo, lo mejor es confiar en este método solo cuando es esencial evaluar el comportamiento de su aplicación contra el hardware real del dispositivo (Google LLC, 2018b).

1.4.2 Frameworks para integración de pruebas

Android Studio puede trabajar en conjunto con diferentes tipos de frameworks para llevar a cabo los dos tipos de prueba, tanto de unidad local que se ejecutan en la máquina virtual de java (JVM) como también pruebas instrumentadas, a continuación, se describirá los frameworks más comunes que se utiliza en cada tipo de prueba, o siguiendo la pirámide de pruebas.

1.4.2.1 Robolectric

Si el entorno de prueba de su aplicación requiere pruebas unitarias para interactuar más extensamente con el marco de Android, se puede usar Robolectric. Esta herramienta ejecuta pruebas lógicas basadas en Java que simulan el marco Android. La comunidad mantiene estos talones. Las pruebas de Robolectric casi coinciden con la fidelidad completa de las pruebas en ejecución en un dispositivo Android mientras aún se ejecutan más rápidamente que las pruebas

en el dispositivo (Google LLC, 2018b). También es compatible con los siguientes aspectos de la plataforma Android:

- Android 4.1 (nivel de API 16) y superior.
- Android Gradle Plugin versión 2.4 y superior.
- Ciclos de vida de los componentes.
- Bucles de eventos.
- Todos los recursos (Google LLC, 2018b).

1.4.2.2 Espresso

Sincroniza las tareas asincrónicas al tiempo que automatiza las siguientes interacciones en la aplicación:

- Realiza acciones en objetos View.
- Completar flujos de trabajo que cruzan los límites del proceso de su aplicación. Disponible solo en Android 8.0 (nivel de API 26) y superior.
- Evaluar cómo los usuarios con necesidades de accesibilidad pueden usar su aplicación.
- Localización y activación de elementos dentro objetos RecyclerViewy AdapterView.
- Validar el estado de los intentos de salida.
- Verificando la estructura de un DOM dentro de los objetos WebView.
- Seguimiento de operaciones de fondo de larga duración dentro de su aplicación (Google LLC, 2018b).

1.4.2.3 UI Automator

El marco UI Automator realiza interacciones dentro de las aplicaciones del sistema en nombre de su aplicación, como inspeccionar la jerarquía de la interfaz de usuario mostrada actualmente, tomar capturas de pantalla y analizar el estado actual del dispositivo. Para obtener más detalles sobre cómo el UI Automator puede observar una aplicación bajo prueba (Google LLC, 2018b).

Como recomendación se debe probar la aplicación usando UI Automator solo cuando su aplicación debe interactuar con el sistema para cumplir con un caso de uso crítico. Debido a que UI Automator interactúa con las aplicaciones del sistema y las IU, debe volver a ejecutar y corregir las pruebas de UI Automator después de cada actualización del sistema. Estas actualizaciones incluyen actualizaciones de la versión de la plataforma Android y nuevas versiones de los servicios de Google Play. Como alternativa al uso de UI Automator,

recomendamos agregar pruebas herméticas o separar su prueba grande en un conjunto de pruebas pequeñas y medianas. En particular, concéntrese en probar una sola comunicación entre aplicaciones a la vez, como enviar información a otras aplicaciones y responder a los resultados de intento (Google LLC, 2018b).

1.4.2.4 Android Test Orchestrator

Android Test Orchestrator ejecuta cada prueba de UI en su propio Instrumentation entorno limitado, lo que aumenta la confiabilidad de su suite de pruebas reduciendo el estado compartido entre las pruebas y aislando las fallas de la aplicación según cada prueba (Google LLC, 2018b).

1.4.2.5 JUnit

Es uno de los frameworks más comunes que usa Android para pruebas de unidad local, este framework está desarrollado para el lenguaje de programación Java, pero Android al trabajar con este lenguaje se puede implementar pruebas que trabajen con JUnit.

Junit es una herramienta de automatización de pruebas unitarias, esta herramienta trabaja en conjunto con el lenguaje de programación JAVA, permitiendo ejecutar grupos o incluso conjunto de grupo de pruebas unitarias, estas pruebas pueden ejecutarse tanto en la máquina virtual de java (JVM) como también en un dispositivo físico.

Uno de los métodos en los cuales se basa JUnit son las aserciones, estos permiten hacer comparaciones entre los distinto tipos de datos que maneja el lenguaje de programación java, como por ejemplo comparar objeto String, Arrays, Objects, así como tipos de datos primitivos enteros, reales, booleanos, etc.

¿Cuál es el objetivo de las pruebas con JUnit?

Uno de los principales objetivos de las pruebas es crear pruebas que conserven su valor a lo largo del tiempo. Alguien más que el autor original tiene que poder ejecutar las pruebas e interpretar los resultados. Debería ser posible combinar pruebas de varios autores y ejecutarlas juntas sin temor a interferencias (Gamma y Beck, 1999, p. 2).

Finalmente, debe ser posible aprovechar las pruebas existentes para crear nuevas. Crear una instalación o accesorio es costoso y un marco tiene que permitir la reutilización de accesorios para ejecutar diferentes pruebas (Gamma y Beck, 1999, p. 2).

Estructura principal de un caso de prueba unitario.

```
public abstract class CasosDePrueba
{
    @Before
    public void setUp() throws Exception
    {
            …
    }

    @After
    public void tearDown() throws Exception
    {
       …
    }

    //(Métodos)casos de prueba
    @Test
    public void activarServicioAccesoRedWifi(){}
    …
    …
 }
```

Los casos de prueba son implementados dentro de clases normales java, dichas clases serán públicas y abstractas, esto con el fin de hacer que esta colección de casos de prueba que se encuentran en dicha clase sea reutilizable por otros casos de prueba, otras clases de casos de prueba o una suite de casos de prueba. Dentro de la clase de Casos de prueba se implementarán los métodos necesarios para cada caso de prueba, así como también métodos que usa por defecto el framework JUnit. Los métodos que por defecto ejecutará Junit serán setUp() y tearDown:

setUp: es el método que se ejecuta antes que cualquier prueba, se utiliza para crear objetos, inicializar variables o realizar cualquier proceso que necesitemos antes de llevar a cabo una prueba, esto tras la definición de la notación (@before → antes)

tearDown: es el método el cual se ejecuta después de haber terminado de ejecutar todos los casos de pruebas, se utiliza normalmente para liberar memoria o dejar de usar servicios, así como también cerrar conexiones a base de datos.

Dentro de las notaciones más destacables en junit se tiene las notaciones **@before→antes** y **@after→después**. Estas dos notaciones son de uso imprescindible a la hora de implementar casos

de prueba ya que, si se requiere que una determinada prueba se ejecute antes o después otra, estás deberán llevar una de estas notaciones antes de la implementación de ese caso de prueba determinado.

JUnit distingue entre fallas y errores. La posibilidad de una falla se anticipa y se verifica con aserciones. Los errores son problemas imprevistos como una ArrayIndexOutOfBoundsException. Las Un AssertionFailedError se desencadena por el método assert proporcionado por TestCase. JUnit proporciona un conjunto de métodos de afirmación para diferentes propósitos (Gamma y Beck, 1999, p. 7).

Ejemplo:

```
protected void assert(boolean condition)
{
        if (! condición)
           throw new AssertionFailedError();
} fallas se señalan con un error AssertionFailedError.
```

Suites de casos de prueba

Las suites de casos de prueba no son más que gestores de clases de pruebas, en estos se decide cuales pruebas se ejecutarán, el orden de estas o incluso se puede tener varias suites de pruebas y gestionar estas a través de otra.

Estructura de una suite de pruebas

```
@RunWith(value= Suite.class)

@Suite.SuiteClasses({
        creacionBDyTabla.class,
        geocodingSW.class,
        filtrarGeocoding.class
    })
```

Dentro de un archivo java se implementará las notaciones necesarias para definir que una clase java funcione como una suite de pruebas. Dentro de la notación @Suite.SuiteClasess({…}) se ubica en orden las clases con las pruebas que se desea que sean ejecutadas, las mismas serán ejecutadas en el orden que se haya definido dentro la esta notación.

Para tener confianza en el estado de un sistema, necesitamos ejecutar muchas pruebas. Hasta este momento JUnit puede ejecutar un solo caso de prueba e informar el resultado en un TestResult.

Nuestro próximo desafío es extenderlo para que pueda ejecutar muchas pruebas diferentes. Este problema se puede resolver fácilmente cuando el invocador de las pruebas no tiene que preocuparse de si ejecuta uno o más casos de prueba. Un patrón popular para sacar en una situación así es Compuesto. Para citar su intención "Componer objetos en estructuras de árbol para representar jerarquías de partes enteras. Compuesto permite a los clientes tratar objetos individuales y composiciones de objetos de manera uniforme". El punto sobre las jerarquías de la parte entera es de interés aquí. Queremos apoyar suites de suites de suites de pruebas (Gamma y Beck, 1999, p. 10).

Jerarquía de organización y ejecución de pruebas con Junit

JUnit tiene una serie de conceptos fundamentales. Un caso de prueba es una clase de Java que prueba algún uso particular de la clase en cuestión. Un caso de prueba consiste en uno o más métodos de prueba, que a su vez prueban algún componente de la clase. Múltiples casos de prueba se pueden combinar en un conjunto de pruebas. Los datos comunes de prueba se pueden organizar en un accesorio, que luego se puede utilizar con los métodos de configuración y desmontaje para aislar la actividad realizada a través de múltiples métodos de prueba dentro de un caso de prueba (Wick et al., 2005, p. 236).

Primeramente, se muestra como está formada la organización de test suites o suites de prueba, mientras que cada suite tiene un número determinado de clase de prueba y a la vez cada una de estas clases de prueba tiene los métodos de prueba de cada caso de prueba específico.

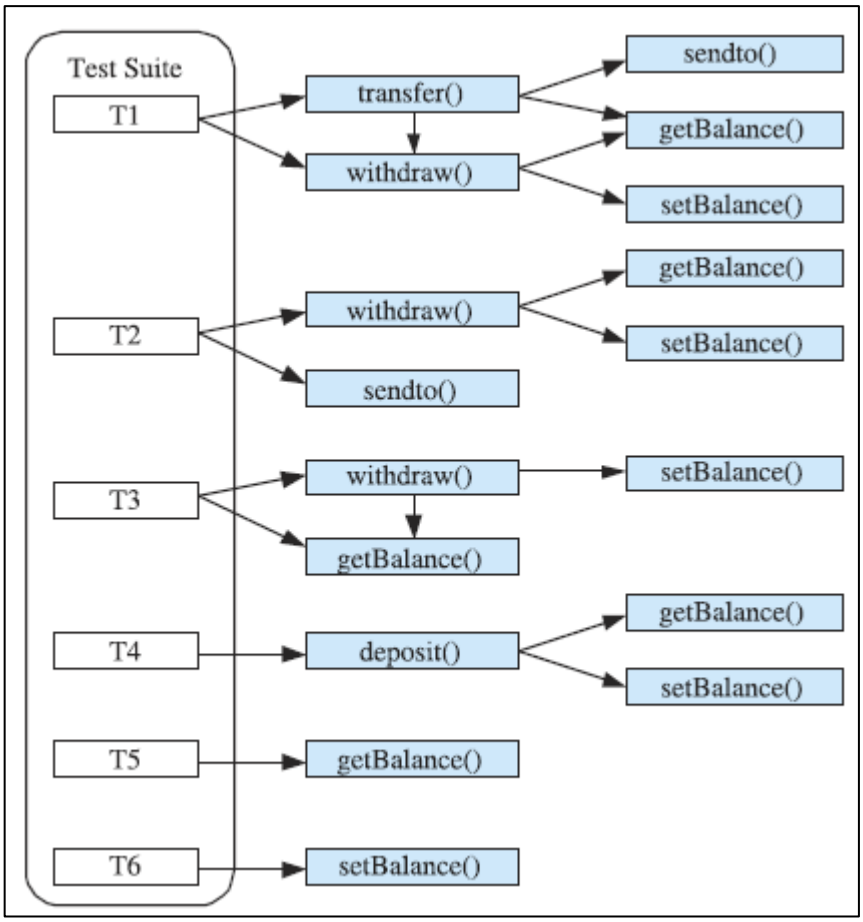


Figura 7-1: Jerarquía de ejecución de casos de prueba.

Fuente: (Mei et al., 2012, p. 1261).

Mientras que en la anterior imagen se pudo apreciar la ejecución y jerarquía de un conjunto de suites de prueba, en la presente imagen se muestra específicamente de una suite para mayor comprensión del funcionamiento de estas.

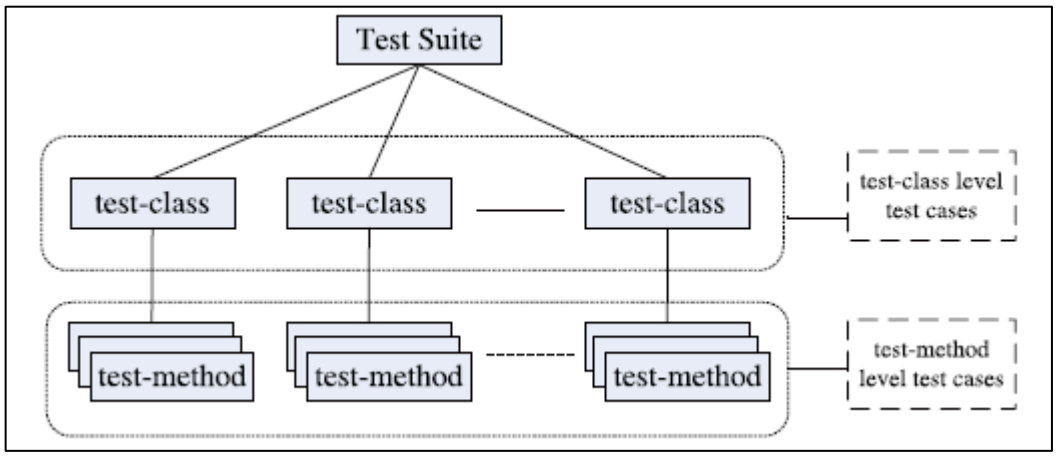


Figura 8-1: Suite de pruebas de software.

Fuente: (Mei et al., 2012, p. 1259).

1.4.3 API Geocoding de Google Maps

La Google Maps Geocoding API es un servicio que proporciona geo codificación convencional e inversa de direcciones (Google LLC, 2018c). La geo codificación es el proceso que convierte direcciones (como la dirección de una calle) en coordenadas geográficas (latitud y longitud) que puedes usar para disponer marcadores en un mapa o posicionar el mapa (Google LLC, 2018c). Google Maps Geocoding API proporciona una forma directa de acceder a esos servicios a través de una solicitud HTTP (Google LLC, 2017b).

La geo codificación inversa es el proceso de conversión de coordenadas geográficas en direcciones en lenguaje natural. El servicio de geo codificación inversa Google Maps Geocoding API también te permite buscar la dirección por un id. de sitio determinado (Google LLC, 2018c).

1.4.3.1 Características

Algunos parámetros son obligatorios y otros opcionales. Como es norma en las direcciones URL, los parámetros se separan con el carácter de Y comercial (&) (Google LLC, 2017b).

Una solicitud de Google Maps Geocoding API debe respetar la siguiente forma:
https://maps.googleapis.com/maps/api/geocode/**outputFormat**?**parameters**

Donde outputFormat puede ser cualquiera de los siguientes valores:

- json (recomendado) indica el formato de salida en JavaScript Object Notation (JSON), o
- xml indica el formato de salida en XML (Google LLC, 2017b).

Parámetros obligatorios en una solicitud de geo codificación:

address: la dirección que quieres geo codificar, en el formato utilizado por el servicio postal nacional del país correspondiente. Se deben evitar elementos de dirección adicionales, como nombres de empresas y números de unidad, habitación o piso (Google LLC, 2017b).

components: un filtro de componente para el que quieres obtener un geo código. Para obtener más información, consulta Filtrado de componentes. El filtro de componentes también se aceptará como un parámetro opcional si se proporciona un parámetro address (Google LLC, 2017b).

key: la clave de API de tu aplicación. Esta clave identifica tu aplicación a los fines de la administración de la cuota (Google LLC, 2017b).

Respuestas de geo codificación

Las respuestas de geo codificación se devuelven en el formato indicado por el marcador output en la ruta de acceso de la dirección URL de la solicitud. Ejemplo, la Google Maps Geocoding API solicita una respuesta json para una consulta sobre "1600 Amphitheatre Parkway, Mountain View, CA" (Google LLC, 2017b).

Esta solicitud muestra el uso del marcador output de JSON:

> https://maps.googleapis.com/maps/api/geocode/json?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&key=YOUR_API_KEY

O esta solicitud muestra el uso del marcador output de XML:

> https://maps.googleapis.com/maps/api/geocode/xml?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&key=YOUR_API_KEY

Figura 9-1: Resultado Json de API geocoding.

Fuente: (Google LLC, 2017b).

La respuesta json contiene dos elementos principales:

- "**status**" contiene metadatos sobre la solicitud. Consulta los siguientes Códigos de estado.
- "**results**" contiene una matriz de información sobre direcciones geo codificadas e información sobre geometría (Google LLC, 2017b).

Se debe tener en cuenta que estos resultados generalmente se deben analizar si quieres extraer valores de ellos.

Códigos de estado

El campo "status" en el objeto de la respuesta de geo codificación contiene el estado de la solicitud y podría contener información de depuración para ayudarte a localizar por qué falló la geo codificación. El campo "status" puede contener los siguientes valores:

"**OK**" indica que no ocurrieron errores, que la dirección se analizó correctamente y que se devolvió al menos un geo código.
ZERO_RESULTS indica que el geo código fue exitoso, pero no devolvió resultados. Esto puede ocurrir si se pasa un valor address inexistente al geo codificador.
"**OVER_QUERY_LIMIT**" indica que excediste tu cuota.
"**REQUEST_DENIED**" indica que se rechazó tu solicitud.
"**INVALID_REQUEST**" generalmente indica que falta la consulta (address, components o latlng).
"**UNKNOWN_ERROR**" indica que no se pudo procesar la solicitud por un error en el servidor. La solicitud puede tener éxito si realizas un nuevo intento (Google LLC, 2017b).

Mensajes de error

Cuando el geo codificador devuelve un código de estado diferente de OK, posiblemente haya un campo error_message en el objeto de respuesta de geo codificación. Este campo contiene información más detallada acerca de los motivos del código de estado proporcionado (Google LLC, 2017b).

Ventajas

- Permite el filtrado de componentes.
- Permite la geo codificación inversa.
- Permite obtener datos de restricción por región.

- Permite la restricción de viewports (Google LLC, 2017b).

Desventajas

Las URL deben estar correctamente codificadas para ser válidas y tienen una limitación de 8192 caracteres para todos los servicios web. Debes tener en cuenta este límite cuando construyas tus URL. Ten en cuenta que los diferentes navegadores, proxies y servidores también pueden tener límites de caracteres diferentes para las direcciones URL (Google LLC, 2017b).

1.5 Herramientas de desarrollo en aplicaciones móviles

1.5.1 Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que aumentan tu productividad durante la compilación de apps para Android (Google LLC, 2017c).

Como las siguientes:

- Un sistema de compilación basado en Gradle flexible
- Un emulador rápido con varias funciones
- Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android Instant Run para aplicar cambios mientras tu app se ejecuta sin la necesidad de compilar un nuevo APK
- Integración de plantillas de código y GitHub para ayudarte a compilar funciones comunes de las apps e importar ejemplos de código
- Gran cantidad de herramientas y frameworks de prueba
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versión, etc.
- Compatibilidad con C++ y NDK
- Soporte incorporado para Google Cloud Platform, lo que facilita la integración de Google Cloud Messaging y App Engine (Google LLC, 2017c).

Estructura del proyecto

Cada proyecto en Android Studio contiene uno o más módulos con archivos de código fuente y archivos de recursos. Entre los tipos de módulos se incluyen los siguientes:

- módulos de apps para Android
- módulos de bibliotecas
- módulos de Google App Engine

De manera predeterminada, Android Studio muestra los archivos de tu proyecto en la vista de proyectos de Android. Esta vista se organiza en módulos para proporcionar un rápido acceso a los archivos de origen clave de tu proyecto (Google LLC, 2017c).

Todos los archivos de compilación son visibles en el nivel superior de Secuencias de comando de Gradle y cada módulo de la aplicación contiene las siguientes carpetas:

manifests: contiene el archivo AndroidManifest.xml.
java: contiene los archivos de código fuente de Java, incluido el código de prueba JUnit.
res: Contiene todos los recursos, como diseños XML, cadenas de IU e imágenes de mapa de bits.

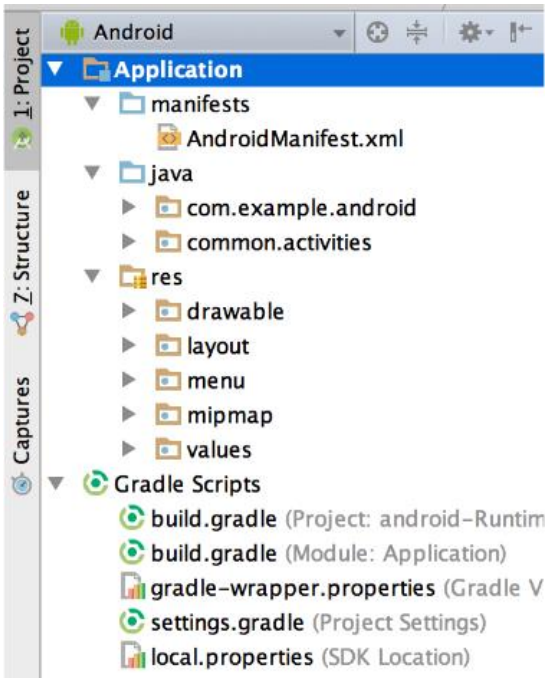


Figura 10-1: Estructura de un proyecto Android.

Fuente: (Google LLC, 2017c).

Interfaz de usuario

La ventana principal de Android Studio consta de varias áreas lógicas que se identifican a continuación.

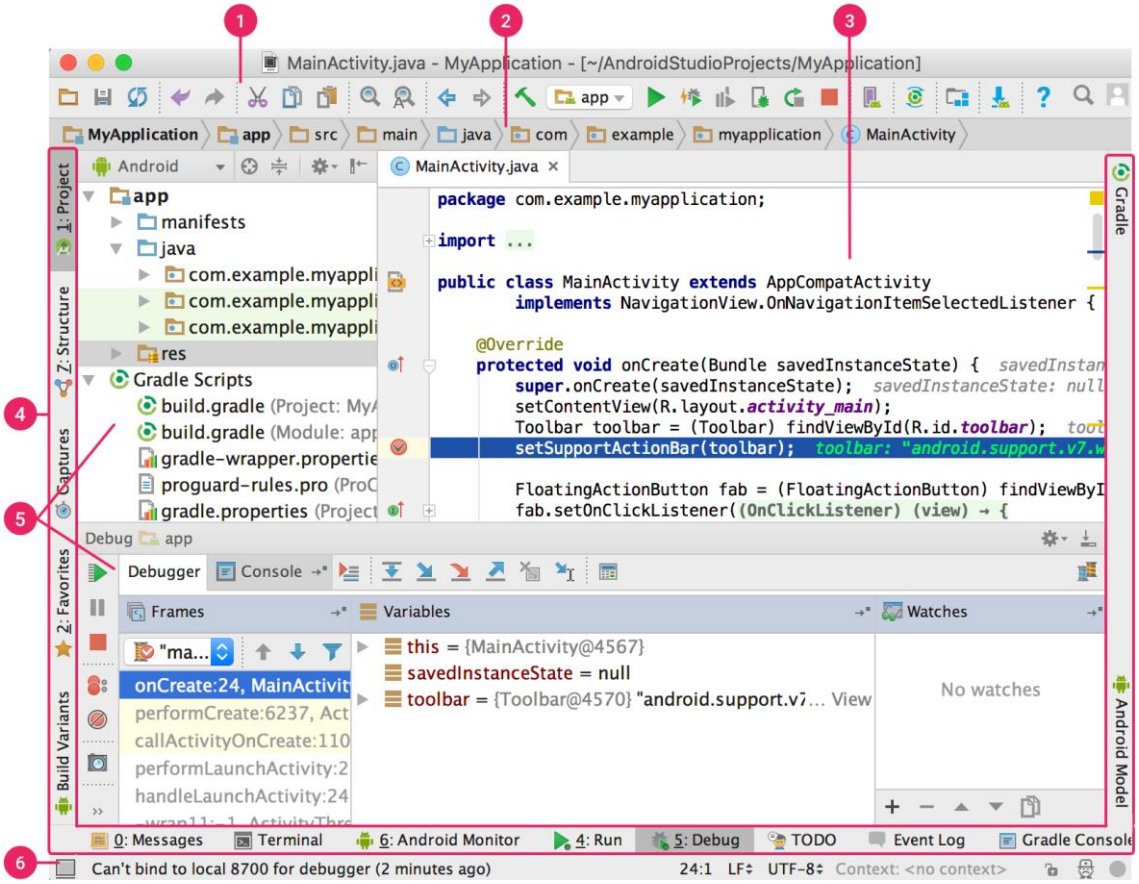


Figura 11-1: Pantalla principal de Android Studio.

Fuente: (Google LLC, 2017c).

1. La barra de herramientas te permite realizar una gran variedad de acciones, como la ejecución de tu app y el inicio de herramientas de Android.
2. La barra de navegación te ayuda a explorar tu proyecto y abrir archivos para editar. Proporciona una vista más compacta de la estructura visible en la ventana Project.
3. La ventana del editor es el área donde puedes crear y modificar código. Según el tipo de archivo actual, el editor puede cambiar. Por ejemplo, cuando se visualiza un archivo de diseño, el editor muestra el editor de diseño.
4. La barra de la ventana de herramientas se extiende alrededor de la parte externa de la ventana del IDE y contiene los botones que te permiten expandir o contraer ventanas de herramientas individuales.
5. Las ventanas de herramientas te permiten acceder a tareas específicas, como la administración de proyectos, las búsquedas, los controles de versión, etc. Puedes expandirlas y contraerlas.
6. En la barra de estado, se muestra el estado de tu proyecto y del IDE en sí, como también cualquier advertencia o mensaje (Google LLC, 2017c).

En cualquier momento, puedes realizar búsquedas en tu código fuente, bases de datos, acciones, elementos de la interfaz de usuario, etc., presionando dos veces la tecla Shift o haciendo clic en la lupa que se encuentra en la esquina superior derecha de la ventana de Android Studio. Esto puede ser muy útil, por ejemplo, si intentas localizar una acción específica del IDE que olvidaste cómo activar (Google LLC, 2017c).

1.5.2 Sublime Text

Es un editor de texto con todas las funciones ideal para editar archivos de texto locales. Tiene muchas funciones incorporadas para ayudar a editar código, como resaltado de sintaxis, sangría automática, reconocimiento de tipo de archivo, una práctica barra lateral de archivo/carpeta para editar fácilmente varios archivos dentro de un directorio, macros para automatizar tareas repetitivas y pestañas y una opción de ventana dividida para ver y editar múltiples archivos al mismo tiempo (Haughee, 2013, p. 3).

Además de las muchas características útiles incorporadas, Sublime Text se construyó desde cero para ser extensible y la comunidad Sublime Text lo ha notado. Ya hay muchas extensiones útiles, incluidas interfaces para sistemas de control de versiones como git, paquetes de fragmentos para jQuery y PHP, y resaltado de sintaxis para lenguajes populares de contenedor CSS como LESS y SASS. Además, la comunidad ha creado Package Manager para que descubrir, instalar, actualizar y administrar estos complementos sea tan simple como escribir una tecla de acceso directo y algunos caracteres. Con un conjunto de funciones sólidas y configurables y la capacidad de extender Sublime Text sin esfuerzo, realmente puede convertirse en su editor de texto y aumentar su productividad de la misma manera que sus herramientas, sin apartarse de su camino (Haughee, 2013, p. 3).

El editor de código sublime text está desarrollado para funcionar en varios sistemas operativos apple, distribuciones de Linux, Windows entre otros, además posee varias características que hace que al momento de programar sea una tarea fácil y más rápida de los normal, a continuación, se presenta las características más sobresalientes de este potente editor.

Características de Sublime Text

- **Mini mapa**

El mini mapa es una característica innovadora que le brinda una vista panorámica del documento que está editando. Siempre presente en el lado derecho del editor, le permite ver rápidamente una

versión actualizada, actualizada y reducida de su documento actual. Si bien el texto rara vez será distinguible, permite una vista topográfica de la estructura de su documento (Haughee, 2013, p. 18).

Figura 12-1: Mini mapa en sublime text.

Fuente:(Haughee, 2013, p. 18).

La función de mini mapa también es muy útil para navegar por un documento grande, ya que puede comportarse de manera similar a una barra de desplazamiento. Cuando se hace clic en, el mini mapa se puede utilizar para desplazar el documento a una parte diferente. Sin embargo, si no necesita el mini mapa, o necesita el espacio real de la pantalla en el que vive, puede ocultarse fácilmente usando la barra de Menú para seleccionar (Haughee, 2013, p. 18).

- **Múltiples cursores**

Otra forma en que Sublime Text se diferencia del abarrotado mercado del editor de texto es incluir la funcionalidad que permite al usuario editar un documento en múltiples lugares al mismo tiempo. Esto puede ser muy útil cuando se realiza un cambio idéntico en varios lugares. Es especialmente útil cuando el cambio que debe ocurrir no se puede realizar fácilmente con Buscar y reemplazar. Cada cursor adicional reflejará el cursor original (Haughee, 2013, p. 19).

- **Modos del editor**

Distracción Free y Vintage Sublime Text ofrece un par de modos distintos. El primero, el modo Distracción Free, crea una interfaz simple para la creación/edición de contenido. El segundo modo

es el modo Vintage. Este modo permite que el editor utilice muchas de las teclas de acceso rápido como Vi/Vim y permite la edición modificada. Ambos se explicarán más a fondo en las siguientes secciones (Haughee, 2013, p. 19).

- **Ir a donde sea (Goto Anything), a un símbolo (Goto Symbol) e ir a una líne (Goto Line)**

Sublime Text hace que sea más fácil navegar por sus proyectos y archivos con varias funciones Goto. Estos se describirán con más detalle en las siguientes secciones (Haughee, 2013, p. 21).

Goto Anything: permite al usuario mostrar rápidamente cualquier archivo abierto. Un cuadro de diálogo mostrará una lista de archivos abiertos, incluidos los archivos en un directorio o subdirectorio que se haya abierto. Esto le permite buscar difusamente un nombre de archivo de cualquier archivo (Haughee, 2013, p. 21).

Goto Symbol: permite la búsqueda difusa de símbolos tales como funciones definidas (Haughee, 2013, p. 22).

Goto Line: Navegar a cualquier línea en el archivo actualmente abierto es tan fácil como presionar Ctrl + G en las tres plataformas admitidas, o invocar Ir a línea desde debajo del elemento Ir a menú, o escribir: (dos puntos) mientras está en Ir a cualquier cosa (Haughee, 2013, p. 23).

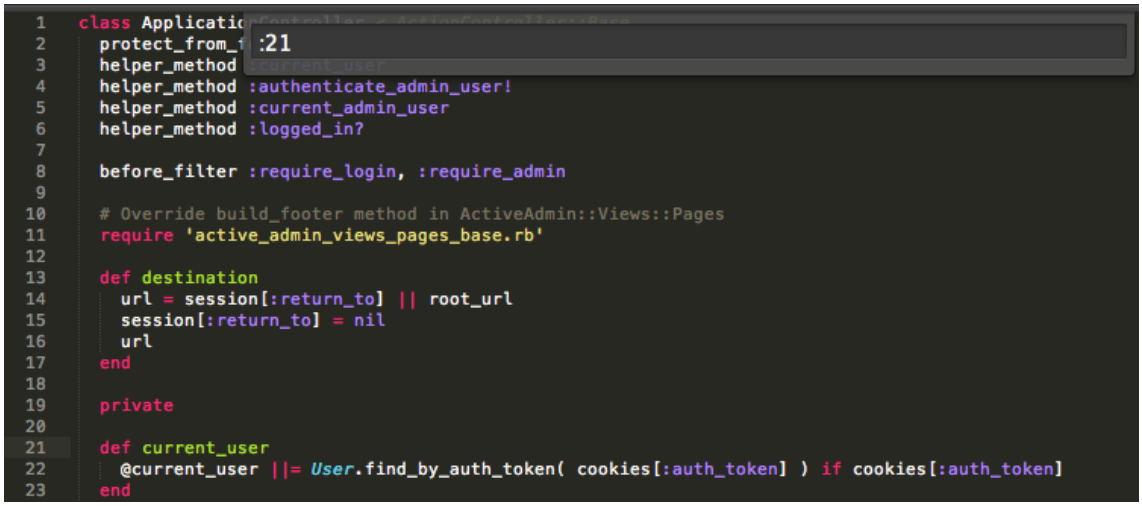


Figura 13-1: Goto Line - Ir a cualquier línea en Sublime Text.

Fuente: (Haughee, 2013, p. 23).

- **Paleta de comandos**

Sublime Text permite que muchos de sus comandos se ejecuten sin salir del teclado a través de la Paleta de comandos (Haughee, 2013, p. 24).

- **Fragmentos**

Sublime Text incluye una función de fragmentos que le permite definir activadores activados por tabulación para expandir el texto. Para crear sus propios fragmentos de texto (Haughee, 2013, p. 27).

1.5.3 Wamp Server

WampServer es un entorno de desarrollo web de Windows. Le permite crear aplicaciones web con Apache2, PHP y una base de datos MySQL. Además, PhpMyAdmin le permite administrar fácilmente sus bases de datos. Este instala automáticamente todo lo que necesita para comenzar a desarrollar aplicaciones web y es muy intuitivo de usar. Podrá sintonizar su servidor sin siquiera tocar los archivos de configuración. Funciona tanto para plataforma de 32 como 64bits (http://www.wampserver.com, 2018).

Características de Wamp Server:

- Una vez que WampServer está instalado, puede agregar manualmente versiones adicionales de Apache, Php o MySql
- El directorio "www" se creará automáticamente (generalmente c: \ wamp \ www), este directorio será donde se almacenen dichos proyectos web.
- Permite administre servicios Apache y MySQL
- Permitir el acceso en línea (online)/fuera de línea (offline) (dar acceso a todos o solo localhost)
- Permite instalar e intercambiar versiones de Apache, MySQL y PHP
- Permite administre la configuración de su servidor
- Permite acceder a los registros, acceder a archivos de configuración y crear alias (http://www.wampserver.com, 2018).

1.6 Gestión de proyectos ágiles – Metodología Scrum

La metodología Scrum para el desarrollo de software en un marco de trabajo diseñado para lograr la colaboración eficaz de equipos en proyectos, que utiliza un conjunto de reglas y artefactos y define roles que generan la estructura necesaria para su correcto funcionamiento (Cadavid et al., 2013, p. 33).

1.6.1 Roles

Los esfuerzos de desarrollo de Scrum consisten en uno o más equipos de Scrum, cada uno compuesto de tres roles de Scrum: Product Owner, Scrum Master y el Development Team. Puede haber otros roles al usar Scrum, pero el marco Scrum solo requiere los tres (Rubin, 2012, p. 14).

Equipo Scrum

Los llamados equipos scrum son autogestionados, multifuncionales y trabajan en iteraciones. La autogestión les permite elegir la mejor forma de hacer el trabajo, en vez de tener que seguir lineamientos de personas que no pertenezcan al equipo y carecen de contexto. Los integrantes del equipo tienen todos los conocimientos necesarios para llevar a cabo el trabajo. La entrega del producto se hace en iteraciones; cada iteración crea nuevas funcionalidades o modifica las que el dueño del producto requiere (Cadavid et al., 2013, p. 33).

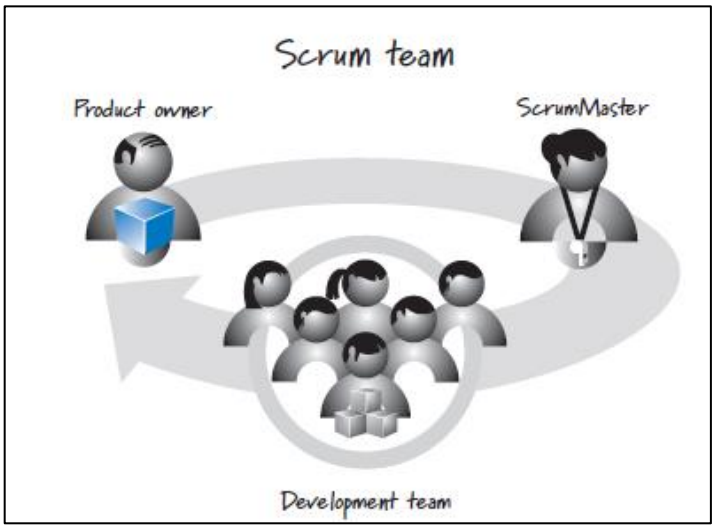


Figura 14-1: Roles dentro de scrum.
Fuente: (Rubin, 2012, p. 15).

El Product Owner o propietario del producto es responsable de lo que se desarrollará y en qué orden. ScrumMaster es responsable de guiar al equipo en la creación y seguimiento de su propio proceso basado en el marco más amplio de Scrum. El Development Team o equipo de desarrollo es responsable de determinar cómo entregar lo que el propietario del producto solicitó (Rubin, 2012, p. 15).

Product Owner: El propietario del producto es el punto central facultado de liderazgo del producto. É1 es la única autoridad responsable de decidir qué características y funcionalidad

construir y el orden en el que construirlas. El propietario del producto mantiene y comunica a los demás participantes una visión clara de lo que el equipo de Scrum está tratando de lograr. Como tal, el propietario del producto es responsable del éxito general de la solución que se desarrolla o mantiene (Rubin, 2012, p. 15).

No importa si el foco está en un producto externo o en una aplicación interna; el propietario del producto todavía tiene la obligación de asegurarse de que siempre se realice el trabajo más valioso posible, que puede incluir trabajo enfocado técnicamente. Para garantizar que el equipo construya rápidamente lo que el propietario del producto desea, el propietario del producto colabora activamente con ScrumMaster y el equipo de desarrollo, y debe estar disponible para responder las preguntas tan pronto como se planteen (Rubin, 2012, pp. 15–16).

ScrumMaster: El ScrumMaster ayuda a todos los involucrados a comprender y adoptar los valores, principios y prácticas de Scrum. Ella actúa como coach, brindando liderazgo en los procesos y ayudando al equipo de Scrum y al resto de la organización a desarrollar su propio enfoque de Scrum, de alto rendimiento y específico de la organización. Al mismo tiempo, Scrum-Master ayuda a la organización a través del desafiante proceso de administración de cambios que puede ocurrir durante la adopción de Scrum (Rubin, 2012, p. 16).

Como facilitador, ScrumMaster ayuda al equipo a resolver problemas y hacer mejoras en el uso de Scrum. También es responsable de proteger al equipo de interferencias externas y asume un papel de liderazgo en la eliminación de impedimentos que inhiben la productividad del equipo. El ScrumMaster no tiene autoridad para ejercer control sobre el equipo, por lo que este rol no es el mismo que el rol tradicional de gerente de proyecto o gerente de desarrollo. El Scrum-Master funciona como un líder, no como un administrador (Rubin, 2012, p. 16).

Development Team: Scrum define el rol de un equipo de desarrollo, que es simplemente una colección diversa y multifuncional de este tipo de personas que son responsables de diseñar, construir y probar el producto deseado (Rubin, 2012, p. 16). Tiene como responsabilidad convertir lo que el cliente quiere, el Product Backlog, en iteraciones funcionales del producto; el equipo de desarrollo no tiene jerarquías, todos sus miembros tienen el mismo nivel y cargo: desarrollador (Cadavid et al., 2013, p. 33).

El equipo de desarrollo se auto organiza para determinar la mejor manera de lograr el objetivo establecido por el propietario del producto. El equipo de desarrollo suele tener entre cinco y nueve personas; sus miembros deben tener colectivamente todas las habilidades necesarias para producir software que funcione bien. Por supuesto, Scrum puede usarse en esfuerzos de desarrollo que

requieren equipos mucho más grandes. Sin embargo, en lugar de tener un equipo de Scrum con, digamos, 35 personas, habría más de cuatro o más equipos de Scrum, cada uno con un equipo de desarrollo de nueve o menos personas (Rubin, 2012, p. 16).

1.6.2 Fases. elementos, actividades y artefactos de Scrum

A continuación, se muestra aquellas actividades y artefactos y la manera que encajan entre sí.

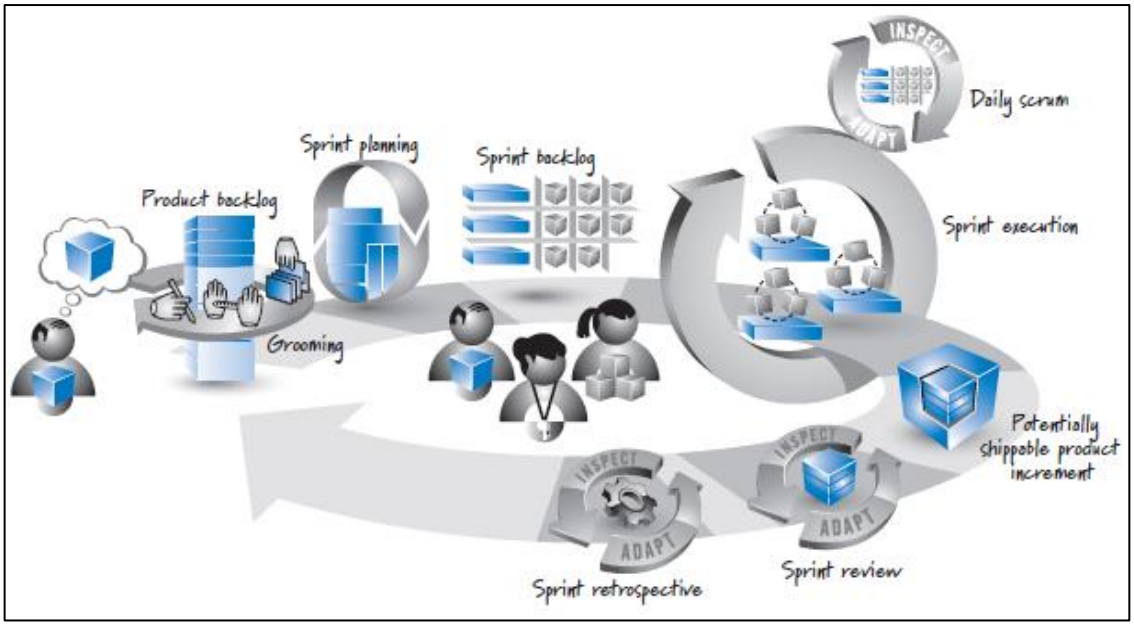


Figura 15-1: Marco de referencia de scrum.

Fuente: (Rubin, 2012, p. 17).

El propietario del producto tiene una visión de lo que quiere crear. Debido a que el cubo puede ser grande, a través de una actividad llamada preparación se divide en un conjunto de características que se recopilan en una lista priorizada llamada product backlog (Rubin, 2012, p. 17).

Un sprint comienza con la planificación de sprints, abarca el trabajo de desarrollo durante el sprint (llamado ejecución sprint) y finaliza con la revisión y retrospectiva. El sprint está representado por una gran flecha que domina el centro de la figura. Es probable que la cantidad de elementos en la cartera de pedidos del producto sea superior a la que un equipo de desarrollo puede completar en un sprint de corta duración. Por esa razón, al comienzo de cada sprint, el equipo de desarrollo debe determinar un subconjunto de los elementos del retraso acumulado del producto que cree que puede completarse, una actividad denominada planificación de sprints, que se muestra justo a la derecha del cubo de acumulación de productos grandes (Rubin, 2012, p. 17).

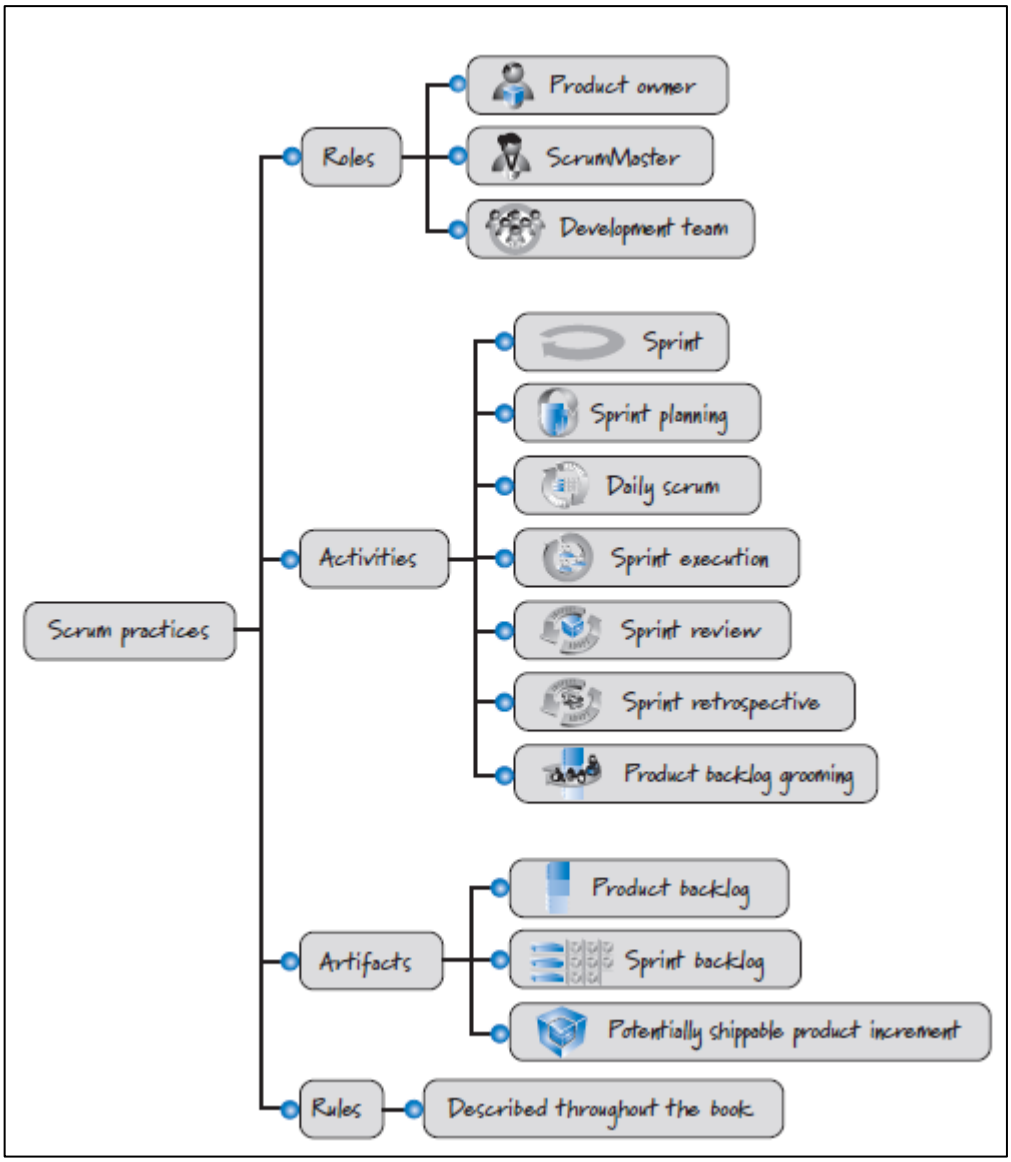


Figura 16-1: Practicas en scrum.

Fuente: (Rubin, 2012, p. 14).

Scrum define un evento principal o Sprint que corresponde a una ventana de tiempo donde se crea una versión utilizable del producto. Cada Sprint es considerado como un proyecto independiente. Su duración máxima es de un mes (Cadavid et al., 2013, p. 33).

1.6.2.1 Product Backlog

La product backlog es el corazón de Scrum. Aquí es donde todo comienza, es básicamente una lista priorizada de requisitos, historias, características o elementos subyacentes. Cosas que el cliente quiere, descritas utilizando la terminología del cliente (Kniberg et al., 2015, p. 9). Se puede decir que es una lista ordenada por valor, riesgo, prioridad y necesidad de los requerimientos que

el dueño del producto define, actualiza y ordena. La lista tiene como característica particular que nunca está terminada, pues evoluciona durante el desarrollo del proyecto (Cadavid et al., 2013, p. 34).

Campos de product backlog

ID: una identificación única, solo un número autoincrementado. Esto es para evitar perder el rastro de las historias cuando las cambiamos de nombre.

Nombre: un nombre breve y descriptivo de la historia. Por ejemplo, "Ver tu propio historial de transacciones". Lo suficientemente claro para que los desarrolladores y el propietario del producto comprendan aproximadamente de lo que estamos hablando, y lo suficientemente claro como para distinguirlo de otras historias. Normalmente 2 - 10 palabras.

Importancia: la calificación de importancia del propietario del producto para esta historia. Por ejemplo 10. O 150. Alto = más importante.

Estimación inicial: la evaluación inicial del equipo de cuánto trabajo se necesita para implementar esta historia en comparación con otras historias. La unidad es puntos de historia y por lo general corresponde aproximadamente a "días-hombre ideales".

Cómo hacer una demostración: una descripción de alto nivel de cómo se demostrará esta historia en la demostración de sprint. Esto es esencialmente una simple prueba de especificación. "Haz esto, luego haz eso, entonces debería suceder" (Kniberg et al., 2015, p. 10).

Usando Scrum, siempre hacemos el trabajo más valioso primero. El propietario del producto, con la opinión del resto del equipo de Scrum y las partes interesadas, es el responsable final de determinar y gestionar la secuencia de este trabajo y comunicarlo en forma de una lista priorizada (u ordenada) conocida como product backlog. En el desarrollo de nuevos productos, los elementos de los pedidos pendientes del producto inicialmente son características necesarias para cumplir con la visión del propietario del producto. Para el desarrollo continuo del producto, el product backlog también puede contener nuevas características, cambios en las funciones existentes, defectos que necesitan reparación, mejoras técnicas, etc. (Rubin, 2012, p. 18).

El propietario del producto colabora con las partes interesadas internas y externas para recopilar y definir los elementos atrasados del producto. Luego se asegura de que los artículos atrasados del producto

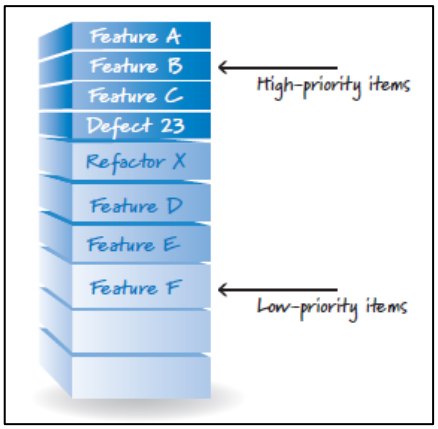


Figura 17-1: Product backlog.

Fuente: (Rubin, 2012, p. 19).

se colocan en la secuencia correcta (usando factores tales como valor, costo, conocimiento y riesgo) para que los artículos de alto valor aparezcan en la parte superior de la cartera de pedidos del producto y los artículos de menor valor aparezcan en la parte inferior. El product backlog es un artefacto en constante evolución. Los artículos pueden ser agregados, eliminados y revisados por el propietario del producto a medida que cambian las condiciones del negocio, o cuando la comprensión del producto del equipo Scrum crece (Rubin, 2012, p. 19).

En general, la actividad de crear y refinar artículos atrasados del producto, estimarlos y priorizarlos se conoce como aseo personal.

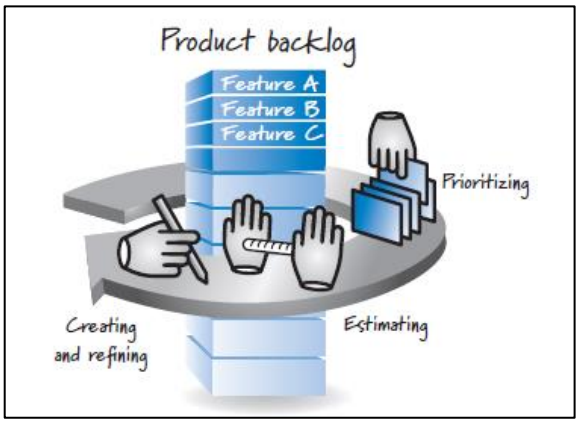


Figura 18-1: Product backlog grooming.

Fuente: (Rubin, 2012, p. 19).

Antes de finalizar la priorización, el pedido o la organización del retraso acumulado del producto, debemos conocer el tamaño de cada artículo en la cartera de pedidos del producto. El tamaño es igual al costo, y los propietarios del producto deben conocer el costo de un artículo para determinar adecuadamente su prioridad. Scrum no dicta qué medida de tamaño usar en caso de

artículos acumulados de productos, si es que los hay. En la práctica, muchos equipos usan una medida de tamaño relativo, como puntos de historia o días ideales. Una medida de tamaño relativo expresa el tamaño total de un artículo de tal manera que no se considera el valor absoluto, pero se considera el tamaño relativo de un artículo en comparación con otros artículos (Rubin, 2012, p. 20).

1.6.2.2 Planificación

En la planificación de sprints se define su plan de trabajo: qué se va a entregar y cómo se logrará. Es decir, el diseño del sistema y la estimación de cantidad de trabajo. Esta actividad dura ocho horas para un Sprint de un mes. Si el Sprint tiene una duración menor, se asigna el tiempo de manera proporcional (Cadavid et al., 2013, p. 33).

La planificación de Sprint es una reunión crítica, probablemente el evento más importante en Scrum. Una reunión de planificación de sprints mal ejecutada puede arruinar todo un sprint.
El objetivo de la reunión de planificación del sprint es brindarle al equipo suficiente información para poder trabajar en paz sin interrupciones durante algunas semanas, y para dar al propietario del producto la suficiente confianza como para permitirle hacerlo (Kniberg et al., 2015, p. 15).

¿Por qué el dueño del producto tiene que asistir?
A veces, los propietarios de los productos son reacios a pasar horas con el equipo haciendo la planificación del sprint. "Chicos, ya he enumerado lo que quiero. No tengo tiempo para estar en tu reunión de planificación ". Ese es un problema bastante serio. La razón por la cual todo el equipo y el propietario del producto tienen que estar en la reunión de planificación del sprint es porque cada historia contiene tres variables que son altamente dependientes entre sí. Estos son: El alcance, la importancia y estimación (Kniberg et al., 2015, pp. 15–16).

El **alcance** y la **importancia** son establecidos por el propietario del producto. La **estimación** es establecida por el equipo. Durante una reunión de planificación de sprints, estas tres variables se ajustan continuamente a través del diálogo cara a cara entre el equipo y el propietario del producto. Normalmente, el propietario del producto comienza la reunión resumiendo su objetivo para el sprint y las historias más importantes. A continuación, el equipo realiza y calcula el tiempo de cada historia, comenzando por la más importante. Mientras hacen esto, se les ocurrirán preguntas importantes sobre el alcance. En algunos casos, las respuestas serán sorprendentes para el equipo, lo que provocará ellos para cambiar sus estimaciones. En algunos casos, el tiempo estimado para una historia no será el esperado por el propietario del producto. Esto puede llevarlo a cambiar la

importancia de la historia. O cambie el alcance de la historia, lo que a su vez causará que el equipo vuelva a estimar, etc. (Kniberg et al., 2015, p. 16).
Este tipo de colaboración directa es fundamental para Scrum y, de hecho, todo el desarrollo de software ágil.

Una product backlog puede representar muchas semanas o meses de trabajo, que es mucho más de lo que se puede completar en un solo sprint corto. Para determinar el subconjunto más importante de elementos atrasados del producto para compilar en el siguiente sprint, el propietario del producto, el equipo de desarrollo y ScrumMaster realizan la planificación del sprint. Durante la planificación del sprint, el propietario del producto y el equipo de desarrollo acuerdan un objetivo de sprint que define lo que se supone que debe alcanzar el próximo sprint. Usando este

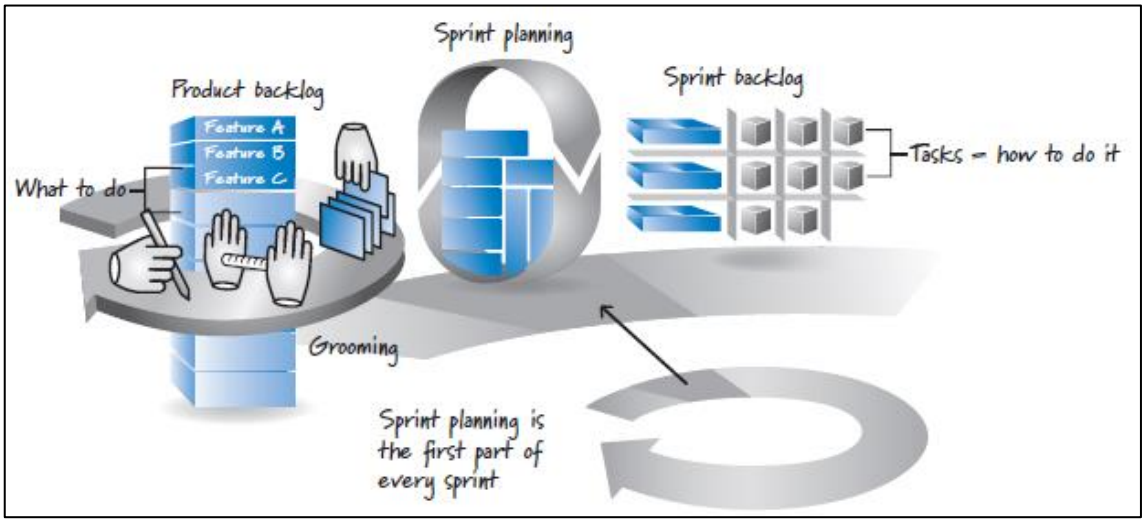


Figura 19-1: Planificación de sprints.
Fuente: (Rubin, 2012, p. 21).

objetivo, el equipo de desarrollo revisa la cartera de productos y determina los elementos de alta prioridad que el equipo puede lograr de manera realista en el próximo sprint mientras trabaja a un ritmo sostenible, un ritmo en el que el equipo de desarrollo puede trabajar cómodamente durante un período prolongado (Rubin, 2012, pp. 21–22).

Luego, el equipo de desarrollo proporciona una estimación (generalmente en horas) del esfuerzo requerido para completar cada tarea. Romper los elementos atrasados del producto en tareas es una forma de diseño y planificación justo a tiempo sobre cómo hacer las funciones (Rubin, 2012, p. 22).

1.6.2.3 Sprint Backlog

El sprint backlog es un subconjunto de ítems del Product Backlog y el plan para realizar en el Incremento del producto. Debido a que el Product backlog está organizado por prioridad, el Sprint backlog es construido con los requerimientos más prioritarios del Product backlog y con aquellos que quedaron por resolver en el Sprint anterior. Una vez construido, el Sprint backlog debe ser aceptado por el equipo de desarrollo, pertenece a éste y solo puede ser modificado por él. Requerimientos adicionales deben ser incluidos en el Product backlog y desarrollados en el siguiente Sprint, si su prioridad así lo indica (Cadavid et al., 2013, p. 34).

El Scrum Master es el encargado de crear el sprint backlog. Esto debe hacerse después de la reunión de planificación del sprint, pero antes del primer scrum diario (Kniberg et al., 2015, p. 45).

En Scrum, el trabajo se realiza en iteraciones o ciclos de hasta un mes calendario llamados sprints. El trabajo completado en cada sprint debe crear algo de valor tangible para el cliente o usuario. Los sprints tienen un intervalo de tiempo, por lo que siempre tienen una fecha de inicio y de finalización fija, y en general todos deben tener la misma duración. Un nuevo sprint sigue inmediatamente a la finalización del sprint anterior. Como regla general, no permitimos ningún cambio de objetivos en el alcance o el personal durante un sprint; sin embargo, las necesidades comerciales a veces hacen que el cumplimiento de esta regla sea imposible (Rubin, 2012, p. 20).

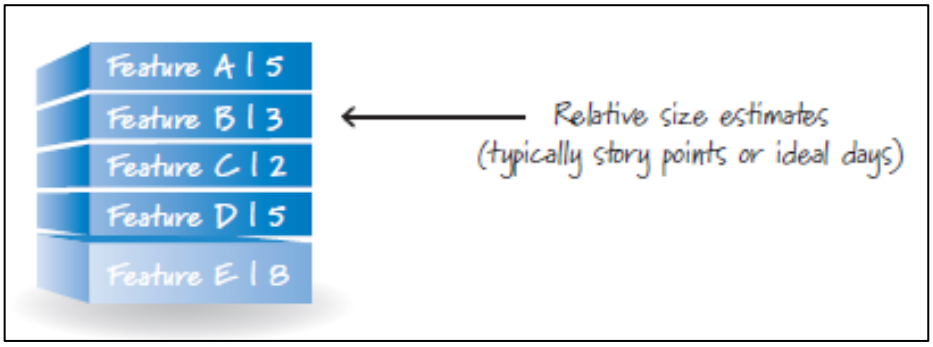


Figura 20-1: Tamaño de items del product backlog.

Fuente: (Rubin, 2012, p. 20).

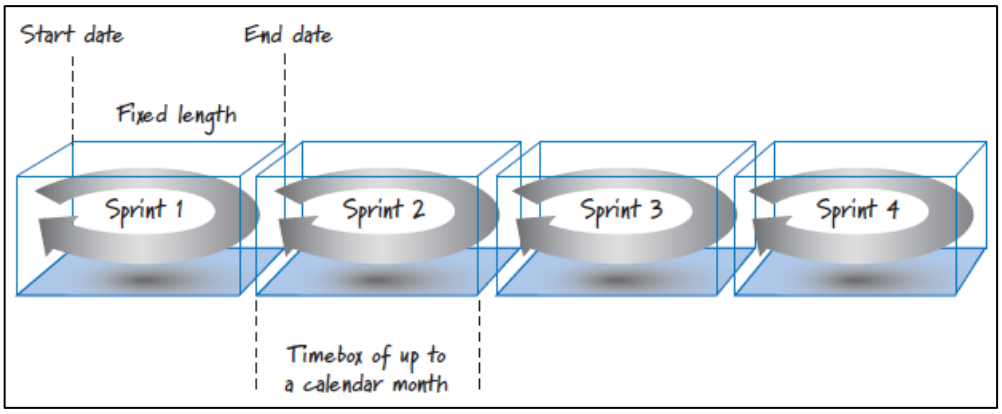


Figura 21-1: Características de sprints.

Fuente: (Rubin, 2012, p. 21).

1.6.2.4 Ejecución de sprints

Una vez que el equipo de Scrum finaliza la planificación de sprint y acuerda el contenido del próximo sprint, el equipo de desarrollo, guiado por el coaching de ScrumMaster, realiza todo el trabajo de nivel de tarea necesario para obtener las características, donde "hecho "Significa que hay un alto grado de confianza en que todo el trabajo necesario para producir características de buena calidad se haya completado (Rubin, 2012, p. 23).

Exactamente qué tareas desempeña el equipo depende, por supuesto, de la naturaleza del trabajo. Nadie le dice al equipo de desarrollo en qué orden o cómo hacer el trabajo a nivel de tarea en la acumulación de sprints. En cambio, los miembros del equipo definen su propio trabajo a nivel de tarea y luego se auto organizan de la manera que consideren mejor para alcanzar el objetivo de sprint (Rubin, 2012, p. 23).

1.6.2.5 Daily scrums

Estas son reuniones diarias que se tiene con el equipo involucrado en el desarrollo del proyecto. Comienzan exactamente a tiempo, todos los días en el mismo lugar. Normalmente se hacen las reuniones de pie, ya que eso reduce el riesgo de superar los 15 minutos (Kniberg et al., 2015, p. 61).

Normalmente en estas reuniones se actualiza la tabla de tareas durante el scrum diario. A medida que cada persona describe lo que hizo ayer y lo hará hoy. Inmediatamente después de la reunión diaria de scrum, alguien resume todas las estimaciones de tiempo y traza un nuevo punto en el sprint burndown (Kniberg et al., 2015, p. 61).

Cada día del sprint, idealmente al mismo tiempo, los miembros del equipo de desarrollo sostienen un scrum diario de tiempo (15 minutos o menos). Esta actividad inspeccionar y adaptar a veces se conoce como el stand-up diario debido a la práctica común de todos los que se levantan durante la reunión para ayudar a promover la brevedad (Rubin, 2012, p. 23).

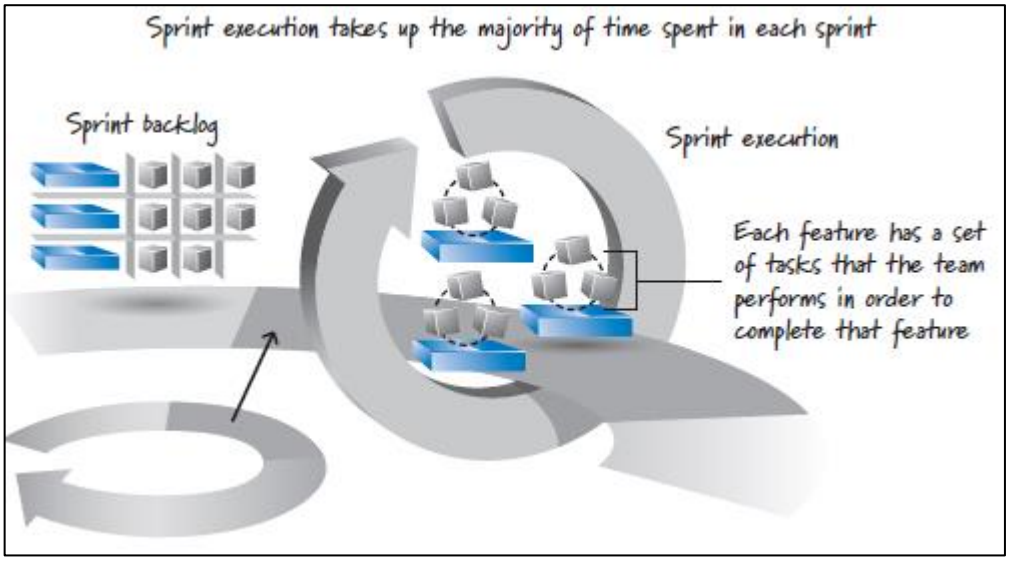


Figura 22-1: Proceso de ejecución de sprint.

Fuente: (Rubin, 2012, p. 23).

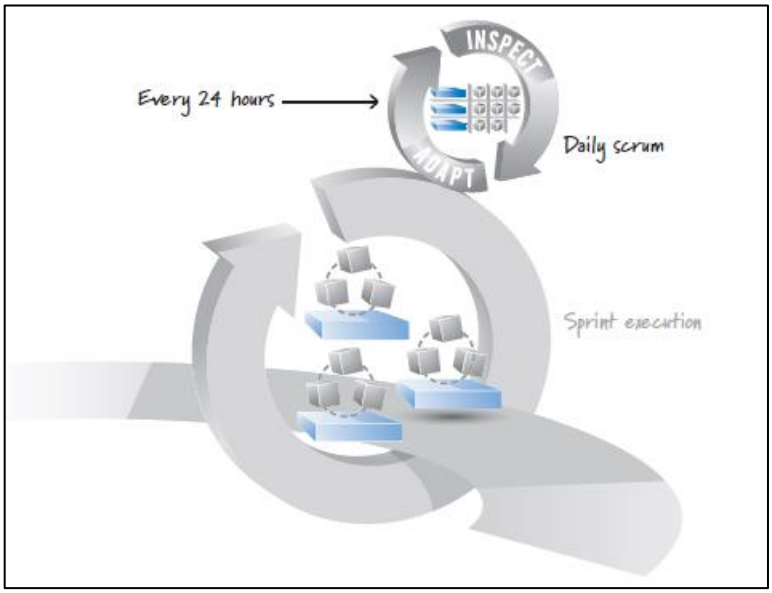


Figura 23-1: Proceso Daily Scrum.

Fuente: (Rubin, 2012, p. 24).

El enfoque común para realizar el scrum diario tiene ScrumMaster facilitador y cada miembro del equipo se turnan para responder tres preguntas en beneficio de los otros miembros del equipo:

- ¿Qué logré desde el último scrum diario?
- ¿En qué planeo trabajar en el próximo scrum diario?
- ¿Cuáles son los obstáculos o impedimentos que me están impidiendo progresar? (Rubin, 2012, p. 24).

Al contestar estas preguntas, todos entienden el panorama completo de lo que está ocurriendo, cómo están progresando hacia el objetivo del sprint, cualquier modificación que quieran realizar en sus planes para el trabajo del día siguiente y qué problemas deben abordarse. El scrum diario es esencial para ayudar al equipo de desarrollo a administrar el flujo de trabajo rápido y flexible dentro de un sprint (Rubin, 2012, p. 24).

El scrum diario no es una actividad de resolución de problemas. Por el contrario, muchos equipos deciden hablar sobre problemas después del scrum diario y lo hacen con un pequeño grupo de personas interesadas. El scrum diario tampoco es una reunión de estado tradicional, especialmente el tipo históricamente llamado por los gerentes de proyecto para que puedan obtener una actualización sobre el estado del proyecto. Sin embargo, un scrum diario puede ser útil para comunicar el estado de los elementos atrasados del sprint entre los miembros del equipo de desarrollo. Principalmente, el scrum diario es una inspección, sincronización y actividad de planificación diaria adaptativa que ayuda a un equipo auto organizador a hacer su trabajo mejor (Rubin, 2012, p. 24).

1.6.2.6 Burndown Chart

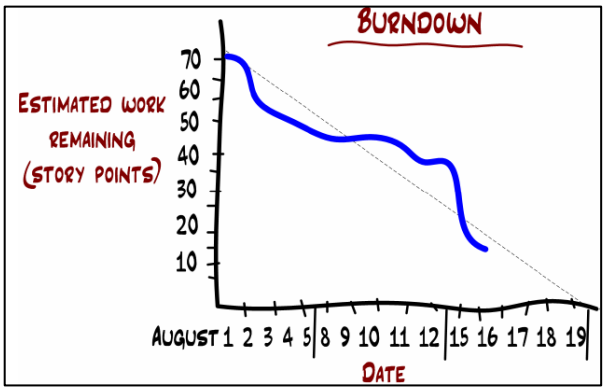


Figura 24-1: Representación Burndown Chart.

Fuente: (Kniberg et al., 2015, p. 51).

Este gráfico nos sirve para ver el avance del proyecto como de va desarrollando entrega tras entrega, sprint tras sprint. La representación gráfica de estos avances por sprint se mostrará en un plano bidimensional en donde el eje x representará las fechas de entrega de cada uno de los sprints,

mientras que en el eje y se representará los puntos estimados que se establecieron para cada una de las entregas. A través de este gráfico podemos darnos cuenta del progreso del proyecto, así como ver ciertas advertencias negativas que se pueden ir presentando y a esto el Scrum Master debe asegurarse que el equipo scrum actúe de buena manera ante estas posibles advertencias.
Un ejemplo claro tenemos cuando existen problemas al subestimar tareas, entonces esto conlleva a que no se cumpla dichas tareas en tiempo estimado. En este caso la solución sería replanificar y remover algunas tareas de los sprints que se encuentran sobrecargados.

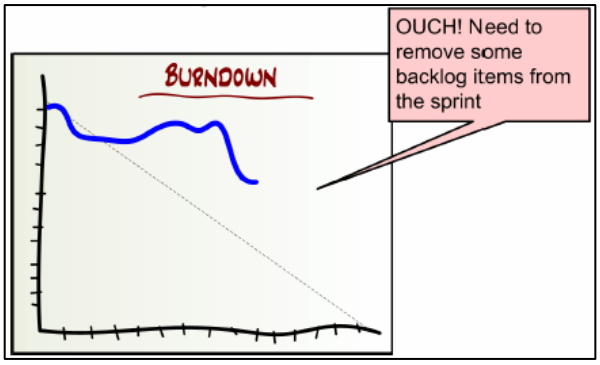


Figura 25-1: Representación Burndown Chart con subestimación de puntos.

Fuente: (Kniberg et al., 2015, p. 52).

Otro de los casos más comunes es la sobrestimación de puntos en las tareas, también en este caso, habrá problemas con el tiempo, ya que tareas que estaban destinadas a terminarse en un tiempo establecido duran menos de lo estimado, dando como resultado tiempo libre que no se hará nada. En este caso se actuaría de igual forma replanificando y agregando más tareas al o a los sprints que se encuentran sobrestimados.

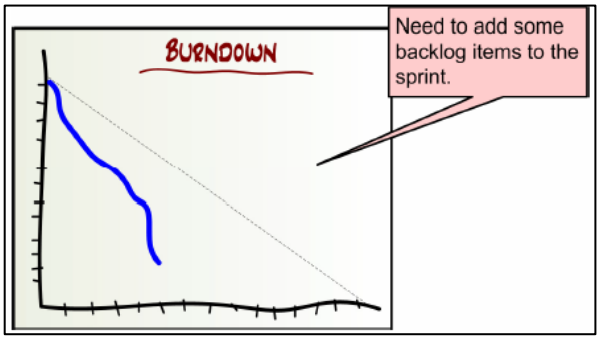


Figura 26-1: Representación Burndown Chart con sobrestimación de puntos.

Fuente: (Kniberg et al., 2015, p. 52).

1.6.2.7 Incremento

Es la suma de todos los ítems terminados en el Sprint backlog. Si hay ítems incompletos deben ser devueltos al Product backlog con una prioridad alta para que sean incluidos en el siguiente Sprint. Se considera que un ítem está terminado si es funcional. La suma de ítems terminados es el producto a entregar (Cadavid et al., 2013, p. 34).

1.6.2.8 Revisión del Sprint

Ocurre al final del Sprint y su duración es de cuatro horas para un proyecto de un mes (o una proporción de ese tiempo si la duración es menor). En esta etapa: el dueño del proyecto revisa lo que se hizo, identifica lo que no se hizo y discute acerca del Product Backlog; el equipo de desarrollo cuenta los problemas que encontró y la manera en que fueron resueltos, y muestra el producto y su funcionamiento. Esta reunión es de gran importancia para los siguientes Sprints (Cadavid et al., 2013, p. 33).

1.6.2.9 Retrospectiva del Sprint

Es una reunión de tres horas del equipo Scrum en la que se analiza cómo fue la comunicación, el proceso y las herramientas; qué estuvo bien, qué no, y se crea un plan de mejoras para el siguiente Sprint. El tiempo, tal como en los casos anteriores, se debe ajustar proporcionalmente en el caso de proyectos de duración menor a un mes (Cadavid et al., 2013, p. 34).

CAPÍTULO II

2 MARCO METODOLÓGICO

2.1 Tipo de investigación

Para llevar a cabo el desarrollo de este proyecto se utilizará la investigación aplicativa, ya que a través de esto podemos hacer uso de nuestros conocimientos adquiridos a lo largo de nuestra carrera universitaria, con el fin de generar nuevos sistemas o a mejorar ya existentes.

2.2 Métodos y técnicas

2.2.1 Metodología SCRUM

Definición

Scrum al ser una metodología de desarrollo ágil tiene como base la idea de creación de ciclos breves para el desarrollo, que comúnmente se llaman iteraciones y que en Scrum se llamarán "Sprints" (Trigas Gallego, 2012, p. 33).

Componentes

Scrum depende de los siguientes elementos como lo son: Product backlog, Historias de usuario, Sprint backlog e Incremento.

- **Product backlog:** Es el corazón de Scrum. Aquí es donde todo comienza. El product backlog es básicamente una lista priorizada de requisitos, historias, características, o lo que sea. Cosas que el cliente quiere, descritas usando la terminología del cliente (Kniberg et al., 2015, p. 9).

  A continuación, se detalla un formato ejemplo de un ítem que pertenece a un producto backlog

  Id: es una identificación única, solo un número auto incrementado. Esto es para evitar perder el rastro de las historias cuando las cambiamos de nombre (Kniberg et al., 2015, p. 9).
  Nombre: nombre corto y descriptivo de la historia. Por ejemplo, "Ver tu propio historial de transacciones". Lo suficientemente claro para que los desarrolladores y el propietario del

producto comprendan aproximadamente de qué estamos hablando, y lo suficientemente claro como para distinguirlo de otras historias. Normalmente 2 - 10 palabras (Kniberg et al., 2015, p. 9).

Importancia: la clasificación de importancia del propietario del producto para esta historia. Por ejemplo 10. o 150. Alto = más importante. Tiendo a evitar el término "prioridad" ya que la prioridad 1 generalmente se considera la prioridad "más alta", lo que se pone feo si luego decides que algo más está incluso más importante. ¿Qué clasificación de prioridad debería obtener eso? ¿Prioridad 0? ¿Prioridad 1? (Kniberg et al., 2015, p. 9).

Estimación inicial: la evaluación inicial del equipo de cuánto trabajo se necesita para implementar esta historia en comparación con otras historias. La unidad es el punto de la historia y por lo general corresponde aproximadamente a "días-hombre ideal" (Kniberg et al., 2015, p. 9).

Demostración: una descripción de alto nivel de cómo se demostrará esta historia en la demostración de sprint. Esto es esencialmente una simple prueba de especificación. "Haz esto, luego haz eso, entonces debería suceder" (Kniberg et al., 2015, p. 10).

Nota: cualquier otra información, aclaraciones, referencias a otras fuentes de información, etc. Normalmente muy breve (Kniberg et al., 2015, p. 10).

PRODUCT BACKLOG (example)					
ID	**Name**	**Imp**	**Est**	**How to demo**	**Notes**
1	Deposit	30	5	Log in, open deposit page, deposit €10, go to my balance page and check that it has increased by €10.	Need a UML sequence diagram. No need to worry about encryption for now.
2	See your own transaction history	10	8	Log in, click on "transactions". Do a deposit. Go back to transactions, check that the new deposit shows up.	Use paging to avoid large DB queries. Design similar to view users page.

Figura 1-2: Ejemplo de Product backlog.

Fuente: (Kniberg et al., 2015, p. 10).

- **Historias de usuario:**

  Son las descripciones de las funcionalidades que va a tener el software. Estas se componen de 3 fases denominadas "Las 3 C":

Tarjeta (Card): Será una breve descripción escrita que servirá como recordatorio.

Conversación (Conversation): Es una conversación que servirá para asegurarse de que se ha entendido bien todo, y concretar el objetivo.

Confirmación (Confirmation): Test funcionales para fijar detalles que sean relevantes e indicar cuál va a ser el límite (Trigas Gallego, 2012, p. 38).

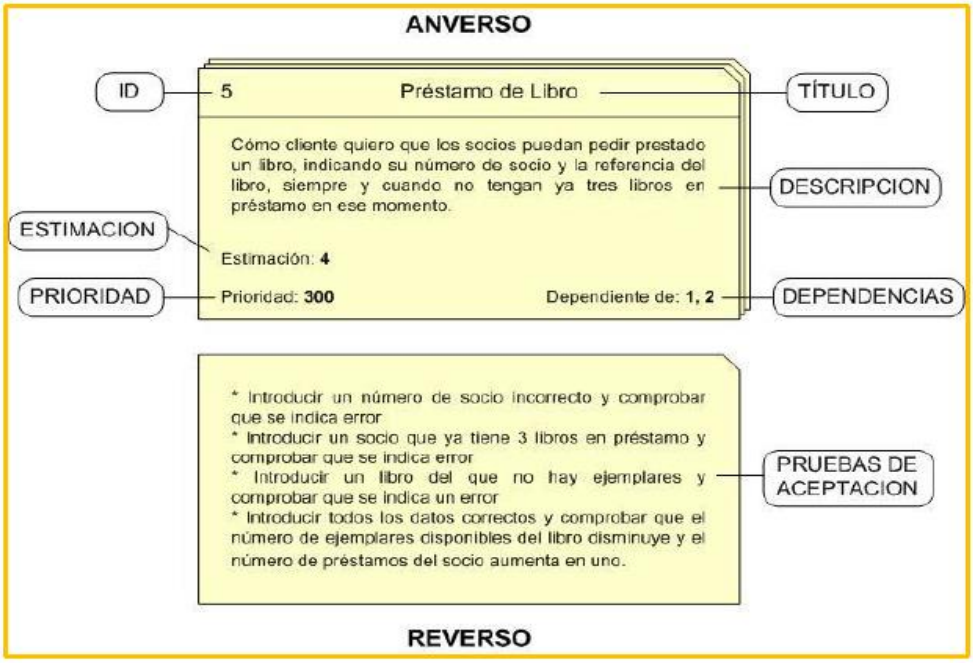


Figura 2-2: Ejemplo de historia de usuario.

Fuente: (Trigas Gallego, 2012, p. 38).

Id: identificador de la historia de usuario

Título: título descriptivo de la historia de usuario

Descripción: Descripción sintetizada de la historia de usuario

Estimación: Evaluación del coste de implementación en unidades de desarrollo. Estas unidades representarán el tiempo teórico (desarrollo/hombre) que se haya estimado al comienzo del proyecto.

Prioridad: en la implementación de la historia de usuario respecto al resto de las historias de usuario. A mayor número, mayor prioridad.

Dependencias: Una historia de usuario no debería ser dependiente de otra, pero a veces es inevitable. En este apartado se indicarán los Ids de las tareas de las que dependen una tarea (Trigas Gallego, 2012, pp. 38–39).

- **Sprint backlog:** El Sprint Backlog es un subconjunto de ítems del Product Backlog y el plan para realizar en el Incremento del producto. Debido a que el Product backlog está organizado

por prioridad, el Sprint backlog es construido con los requerimientos más prioritarios del Product backlog y con aquellos que quedaron por resolver en el Sprint anterior. Una vez construido, el Sprint backlog debe ser aceptado por el equipo de desarrollo, pertenece a éste y solo puede ser modificado por él. Requerimientos adicionales deben ser incluidos en el Product backlog y desarrollados en el siguiente Sprint, si su prioridad así lo indica (Cadavid et al., 2013, p. 34).

- **Incremento:** El Incremento es la suma de todos los ítems terminados en el Sprint backlog. Si hay ítems incompletos deben ser devueltos al Product backlog con una prioridad alta para que sean incluidos en el siguiente Sprint. Se considera que un ítem está terminado si es funcional. La suma de ítems terminados es el producto a entregar (Cadavid et al., 2013, p. 34).

Ciclo de vida de un proyecto.

Existe un ciclo de desarrollo correspondiente a 30 días, llamado Sprint, el cual es precedido por las actividades de las fases Pre-juego y Post-juego. Adicionalmente, el objetivo del Sprint es establecido, el cual sirve como un criterio mínimo para el éxito del mismo y mantiene al equipo enfocado en panorama global y no estrictamente en tareas específicas (Delgadillo et al., 2016, p. 151). Se definen tres reuniones para el Sprint, la primera es una junta (30 minutos) que permite al equipo monitorear el estatus y los problemas de comunicación del proyecto, la segunda se realiza para mostrar el avance al Product Owner para obtener una observaciones, dudas o cambios sobre el producto, la tercera es una retroalimentación sobre lo bueno y lo malo de proceso utilizado en el anterior ciclo y se propone actividades de mejora (Delgadillo et al., 2016, p. 151).

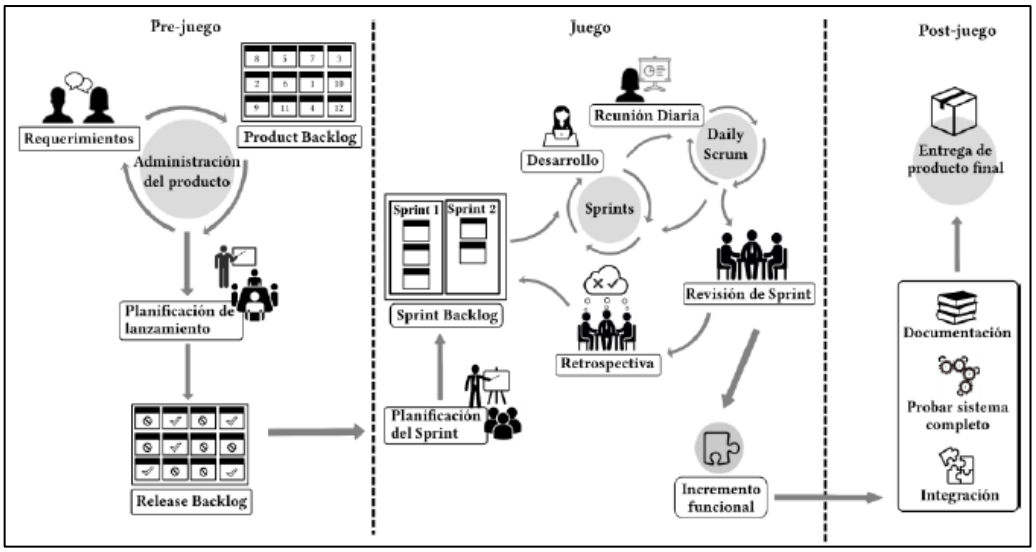


Figura 3-2: Ciclo de vida de Scrum.

Fuente: (Delgadillo et al., 2016, p. 151).

2.3 Técnicas

Observación: Con esta técnica se podrá ver la conducta de la aplicación de las pruebas automáticas de software sobre la aplicación móvil.

Revisión de documentación: Como manera de guía en el desarrollo del proyecto se obtendrá información de fuentes fidedignas como libros, foros, papers, artículos y revistas científicas, tesis relacionadas al proyecto, las cuáles serán tomadas como referencia.

2.4 Modelos tipológicos propuestos con base en metodología Scrum

Haciendo énfasis en que la metodología ágil scrum no tiene modelos globales o universales en los que todos y cada uno de los proyectos se basan, sino más bien cada empresa o incluso cada proyecto definen sus propios modelos o formatos de documentos, con el fin de llevar a cabo el seguimiento del proyecto con esta metodología. Por lo que a continuación se presenta la propuesta con la cual se hará el seguimiento de este proyecto de titulación.

Modelo de Product Backlog

En este modelo se especifica todos los requerimientos tanto funcionales como los técnicos, dando una estimación en base a puntos acordados y asignando una prioridad esto decidido por el Product Owner.

Id: Identificador único el cual seguirá el siguiente formato: tendrá las abreviaciones de HU (Historia de usuario) o HT (Historia técnica) seguido por un guion bajo que a continuación, tendrá el número asignado a tal historia con una longitud de dos dígitos Ejemplo: HU_10

Requerimientos: Lleva una descripción breve de cada uno de los requerimientos que se realizarán en dicha historia.

Prioridad: La prioridad de cada historia se ha definido como alta, media y baja, las cuales serán asignadas dependiendo la historia y a decisión del Product Owner.

Puntos estimados: Los puntos estimados serán definidos en base a la cantidad de horas dedicadas al proyecto en una unidad de sprint, así que un sprint será de dos semanas laborables en la cual cada día laborable es de 5 puntos. Lo que lleva a un total de 50 puntos por cada sprint.

Tabla 1-2: Ejemplo modelo de product backlog.

HISTORIAS TÉCNICAS/ DE USUARIO			
ID	**Requerimientos**	**Prioridad**	**Puntos Estimados**
HU_01	Consumo de servicio web método post	Alta	10
HT_01	Instalación de herramienta Android Studio	Media	30
…	…	Baja	…

Realizado por: Wilmer B. 2018.

Modelo de Sprint Backlog

En el siguiente modelo se realiza la planificación inicial del proyecto, en el mismo se detalla los siguientes campos:

Sprint: Se ubica el número correspondiente al sprint.
Id: Es el identificador único ya sea de una historia de usuario o historia técnica
Esfuerzo: Son aquellos puntos en los que se estimó cada una de las tareas ya antes establecidas
Esfuerzo Total: Es la suma de todos los puntos de esfuerzo correspondientes a un sprint
Tipo: Corresponde a la fase en la que se encuentra dicha historia, pudiendo estar estas en la fase de: Análisis, Diseño, Desarrollo, Documentación o Pruebas.
Fecha Inicio: Es la fecha en la que se inicia a desarrollar dicha historia.
Fecha Fin: Es la fecha en la que se termina de desarrollar dicha historia.

Tabla 2-2: Ejemplo modelo de sprint backlog.

Sprint	ID	Esfuerzo	Esfuerzo Total	Tipo	Fecha Inicio	Fecha Fin
1	HU_01	10	50	Análisis	19/03/17	20/03/17
	HU_02	40		Diseño	21/03/17	30/03/17
2	HT_01	30	50	Desarrollo	…	…
	HU_03	20		Prueba	…	…
…	…	…	…	Documentación	…	…

Realizado por: Wilmer B. 2018.

Modelo de Historia de usuario o Historia técnica

En el siguiente modelo se detalla los campos que definen a una historia de usuario, la misma que será aplicada a cada historia d usuario o técnica.

Id: Es el identificador único que corresponde a dicha historia, ya sea (HT o HU).

Nombre: El nombre o la descripción que fue establecida en el product backlog correspondiente a dicha historia.

Descripción: Es una breve descripción de ¿Por qué? y ¿Para qué? Va a realizar dicha historia.

Usuario: Es aquella persona que utilizará la funcionalidad en caso de ser una historia de usuario y en caso de ser una historia técnica el usuario será la persona encargada en desarrollar la misma.

Sprint: Es el número de sprint al que corresponde dicha historia.

Fecha Inicio: Es la fecha de inicio de desarrollo definida en el sprint backlog para tal historia.

Fecha Fin: Es la fecha de fin de desarrollo definida en el sprint backlog para tal historia.

Esfuerzo: Es la cantidad de puntos que tomó realizar dicha historia de usuario.

Tareas de Ingeniería: En este apartado se enumeran las tareas de ingeniería que corresponden a la historia correspondiente, los campos definidos para localizar dichas tareas son:

- **Id:** Es el identificador único que se asigna a una tarea de ingeniería, este empieza con las abreviaciones TI, separado por un guion bajo del número de tarea que es, dependiendo del sprint al que corresponde. Este número secuencial se vuelve a iniciar en 01 en cada historia de usuario. Ejemplo TI_01
- **Nombre:** Es la descripción de la tarea que se realizará para dicho sprint.

Tabla 3-2: Ejemplo modelo historia de usuario/técnica.

<table>
<tr><th colspan="4">HISTORIA DE (USUARIO/TÉCNICA)</th></tr>
<tr><td colspan="2">Id: (HT/HU) _01</td><td colspan="2">Nombre: Consumo de servicio web método post</td></tr>
<tr><td colspan="4">Descripción: Se realizará la implementación de la funcionalidad para consumir el servicio web del servicio de hosting para la sincronización de base de datos.</td></tr>
<tr><td colspan="2">Usuario: Wilmer Barrera</td><td colspan="2">Sprint: 1</td></tr>
<tr><td colspan="2">Fecha inicio: 19/03/17</td><td>Fecha fin: 20/03/17</td><td>Esfuerzo: 10</td></tr>
<tr><th colspan="4">TAREAS DE INGENIERÍA</th></tr>
<tr><th>Id</th><th colspan="3">Nombre</th></tr>
<tr><td>TI_01</td><td colspan="3">Realizar conexión para petición http en Android</td></tr>
<tr><td>…</td><td colspan="3">…</td></tr>
</table>

Realizado por: Wilmer B. 2018.

Modelo de Tarea de ingeniería

Para llevar a cabo las tareas de ingeniería correspondientes a cada historia se lo aplicará bajo el siguiente formato:

Id: Es el identificador que anteriormente fue asignada a dicha tarea en la historia correspondiente.
Tipo de tarea: se especifica la fase a la que corresponde dicha tarea, esta puede ser análisis, diseño, desarrollo o pruebas. Normalmente las tareas de usuario se realizan en las fases intermedias antes ya descritas.
Sprint: Corresponde al número de sprint en el que se encuentra la historia de usuario a la cual pertenece dicha tarea.
Nombre de historia de (usuario/técnica): en este campo se debe poner primero el identificador al cual pertenece dicha tarea de ingeniería, seguido de este el nombre que tiene dicha historia.
Nombre tarea: En este campo se asigna el nombre ya antes descrito en la tabla de historia de usuario, el cual corresponde a la tarea de ingeniería.
Fecha de inicio: Es la fecha en la que comienza a realizarse dicha tarea de ingeniería
Fecha Fin: Es la fecha en la que termina de realizarse dicha tarea de ingeniería.
Descripción: Se detalla los funcionalidades o métodos que se realizarán para llevar a cabo dicha tarea de ingeniería.
Pruebas de aceptación: En este apartado se enumeran una lista de pruebas de aceptación por las cuales tendrá que pasar dicha tarea de ingeniería. Las mismas se encuentran definidas por los siguientes campos:

- **Id:** Es el identificador establecido para localizar cada prueba de aceptación, este id estará formado por las abreviaciones (PA) seguido por un guion bajo y a continuación un número secuencial el cual deberán seguir todas las pruebas de aceptación. Ejemplo PA_01.

- **Nombre:** Es una breve descripción de la prueba que se realizará en tal tarea de ingeniería. Este campo se encuentra descrito ya anteriormente en la tarea de ingeniería correspondiente.

Tabla 4-2: Ejemplo modelo tarea de ingeniería.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 1
Nombre de historia (usuario/técnica): HU_01 Consumo de servicio web método post		
Nombre tarea: Realizar conexión para petición http en Android		
Fecha inicio: 19/03/17		**Fecha fin:**19/03/17
Descripción: - Creación de método para validar tipo de conexión wi-fi o móvil - Creación de consultas a la base de datos - …		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_01	Verificar que exista conexión a internet
…	…

Realizado por: Wilmer B. 2018.

Modelo de Prueba de aceptación manual (personal humano)

El presente modelo sirve de guía para detallar las características que se toman en cuenta dentro de una prueba de software con personal humano.

Formato de pruebas de aceptación manuales:

Id: Es el identificador el cual fue asignado ya anteriormente en la tabla de la tarea de ingeniería correspondiente.
Nombre: Dicho campo es el mismo que se encuentra definido ya anteriormente en la tabla de la tarea de ingeniería correspondiente a la actual prueba.
Historia de (Usuario/Técnica): En este campo se ubica primero el identificador de la historia, seguido por el nombre de la historia a la cual pertenece.
Responsable: Es la persona encargada en realizar dicha prueba de aceptación.
Fecha: Es la fecha en la cual se realizó dicha prueba.
Descripción: Debe ser una descripción que defina ¿Por qué? Es necesario hacer dicha prueba.
Condición de ejecución: Son las tareas o requisitos que se deben tener en cuenta antes de realizar dicha prueba.
Pasos de ejecución: Se detalla el paso a paso que toma realizar la prueba. Se pueden agregar los datos con los que se realizó dicha prueba de ser necesario.
Resultado esperado: Corresponde a como se espera que reaccione la funcionalidad ante la prueba aplicada, se detalla de manera específica.
Evaluación de la prueba: Es el estado que se obtuvo después de aplicar la prueba, esta puede ser exitosa o fallida.
Tiempo dedicado: Cada uno de los responsables de realizar las pruebas tomarán tiempos cronometrados desde el inicio hasta el fin de la prueba. El mismo tiempo servirá para una comparación de resultados tanto de aplicación de pruebas de aceptación manuales como automáticas.

Tabla 5-2: Ejemplo modelo prueba de aceptación.

PRUEBA DE ACEPTACIÓN	
Id: PA_01	**Nombre:** Verificar que exista conexión a internet
Historia de (usuario/técnica): HU_01 Consumo de servicio web método post	
Descripción: Para consumir el servicio web post se necesita una conexión a internet	
Responsable: Wilmer Barrera	**Fecha:** 18/11/17
Condición de ejecución: - Los servicios web a ser consumidos deben estar creados - La conexión de petición http debe estar creada - ….	
Pasos de ejecución: - Instalar la app móvil - Activar datos móviles o conectarse a una red wi-fi - Entrar a la app móvil - Entrar a la pantalla de sincronización - Dar clic en el botón sincronizar	
Resultado esperado: Cuando se da clic en el botón sincronizar, y en caso de que hay internet mostrará un mensaje de sincronización exitosa, caso contrario mostrará un mensaje de conexión fallida,	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 33 segundos

Realizado por: Wilmer B. 2018.

2.5 Estudio preliminar

La empresa Intercompu-Technology es una empresa de la ciudad de Riobamba, la misma tiene un objetivo futuro con el cual desea brindar información acerca de horarios de rutas específicas de los transportes tanto públicos como privados. Debido a que cada empresa o cooperativa de transporte muestran información únicamente de esta, lo que hace que las personas deban acudir a los terminales para poder informarse de todos los horarios que son cubiertos por dichas rutas o también deben entrar a cada uno de los sitios web de las cooperativas en caso de existir.

Pero lo que se pretende con la implementación de la aplicación móvil es unificar esta información correspondiente a las cooperativas afiliadas a la empresa Intercompu-Technology. Esto con el fin de brindar un mejor servicio a las personas que comúnmente viajan a las distintas partes del Ecuador.

Uno de los principales procesos a la hora de agregar calidad a un producto software, este se los realiza en la fase de desarrollo, por lo que es un recurso indispensable a la hora de hacer que un producto software sea confiable para un usuario, mientras que en la mayoría de empresas pequeñas dedicadas a la producción de software utilizan o aplican mal este proceso debido a que las mismas personas que desarrollan el software son las que prueban el sistema o el método de aprobación de una prueba es en base a las pruebas de aceptación lo que en sí es el criterio personal de alguien.

Con la automatización de pruebas en el sistema se pretende cubrir todas las brechas posibles en las que se den resultados tanto positivos como negativos, y así mismo ver cómo reaccionará el sistema antes esto.

2.6 Fase de planificación

La presente fase propone una planificación en la cual se desarrollará cada una de las tareas planificadas. Estas han sido organizadas en orden de prioridad requeridas por el cliente, las mismas se la ha gestionado con un software libre de código abierto con nombre GanntProject. Dicha planificación se ha representado mediante un diagrama Gantt el cual se encuentra en el **Anexo A.** Planificación de tareas.

El presente proyecto se realizará con una duración de 100 días laborables y un total de 500 puntos los cuales se han cumplido acorde a lo planificado.

2.6.1 Planificación inicial

A continuación, se presentará la planificación siguiendo el formato de Product Backlog definido en la **Tabla 1-2** siguiendo la metodología scrum, la cual se ha realizado en base a la recolección de requerimientos con el cliente.

El trabajo y reuniones en conjunto con el personal que forma parte de dicho proyecto se han establecido un total de 18 tareas funcionales y 13 tareas no funcionales, las cuales se muestran a continuación.

Tabla 6-2: Product Backlog - Historias de Usuario.

HISTORIAS DE USUARIO			
ID	**Requerimientos**	**Prioridad**	**Puntos Estimados**
HU_01	Codificación de scripts SQL para transacciones en base de datos	Alta	40
HU_02	Implementación de interfaz para consulta de horarios por cooperativa de transporte	Baja	10
HU_03	Implementación de servicios web Geocoding API de Google Maps a través de petición GET	Media	25
HU_04	Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps	Alta	15
HU_05	Implementación de interfaz para consulta de horarios en base a ciudad origen y destino	Baja	5
HU_06	Implementación de funcionalidad para consulta de horarios por transporte	Media	20
HU_07	Implementación de interfaz favoritos	Alta	10
HU_08	Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino	Alta	25
HU_09	Implementación de funcionalidad para obtener la próxima salida en base a la hora actual	Alta	10
HU_10	Creación de servicios web en PHP para sincronización con app móvil	Alta	25
HU_11	Implementación de funcionalidad para mostrar horarios por días	Media	20
HU_12	Implementación de funcionalidad para obtener coordenadas geográficas en Android	Alta	15
HU_13	Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte	Alta	25
HU_14	Implementación de funcionalidad favoritos	Alta	25
HU_15	Implementación de funcionalidad llamada telefónica y cargar datos de agencia de cooperativa	Baja	15
HU_16	Consumo de servicios web a través de petición POST	Alta	5
HU_17	Sincronización de app móvil con base de datos alojado en servicio de hosting	Alta	30
HU_18	Implementación de interfaz de instalación	Alta	20

Realizado por: Wilmer B. 2018.

Tabla 7-2: Product Backlog - Historias Técnicas.

HISTORIAS TÉCNICAS			
ID	**Requerimientos**	**Prioridad**	**Puntos Estimados**
HT_01	Análisis de herramientas a emplear	Alta	10
HT_02	Análisis y gestión de riesgos	Baja	15
HT_03	Diseño de base de datos	Media	25
HT_04	Diseño de arquitectura de la aplicación móvil	Alta	5

HT_05	Desarrollo de manual de usuario	Media	30
HT_ 06	Implementación de pruebas automatizadas Sprint 2	Media	10
HT_07	Implementación de pruebas automatizadas Sprint 3	Media	10
HT_08	Implementación de pruebas automatizadas Sprint 4	Media	10
HT_09	Implementación de pruebas automatizadas Sprint 5	Media	5
HT_10	Implementación de pruebas automatizadas Sprint 6	Media	5
HT_11	Implementación de pruebas automatizadas Sprint 7	Media	10
HT_12	Implementación de pruebas automatizadas Sprint 8	Media	10
HT_13	Implementación de pruebas automatizadas Sprint 9	Media	15

Realizado por: Wilmer B. 2018.

2.6.2 Personas y roles de usuario

Dentro del seguimiento y colaboración del personal en el presente proyecto se ha definido los siguientes roles para cada persona.

Tabla 8-2: Personas y roles del proyecto.

Personas	Rol
Ing. Raúl Rosero	Scrum Master
Ing. Richard Guamán	Propietario del producto
Wilmer Barrera	Developer

Realizado por: Wilmer B. 2018.

2.6.3 Plan de entrega

Con los requerimientos recolectados y representados mediante el **product backlog**, el siguiente paso es realizar el proceso de **sprint backlog**, en el cual se planifican dichas entregas o avances de proyecto estos han sido establecidos en conjunto con el propietario del producto. Los cuales se ha llegado a un acuerdo para las diferentes entregas de versiones del sistema, las tareas han sido planificadas por sprints, este sprint consta de dos semanas laborables con una duración de 5 horas diarias reuniendo un total de 50 horas por cada sprint.

Tabla 9-2: Sprint Backlog.

Sprint	ID	Esfuerzo	Esfuerzo Total	Tipo	Fecha Inicio	Fecha Fin
1	HT_01	10	50	Análisis	02/10/17	03/10/17
	HT_02	15		Análisis	04/10/17	06/10/17
	HT_03	25		Diseño	09/10/17	13/10/17
2	HU_01	40	50	Desarrollo	16/10/17	25/10/17

	HT_06	10		Pruebas	26/10/17	27/10/17
3	HT_04	5	50	Diseño	30/10/17	30/10/17
	HU_02	10		Desarrollo	31/10/17	01/11/17
	HU_03	25		Desarrollo	02/11/17	08/11/17
	HT_07	10		Pruebas	09/11/17	10/11/17
4	HU_04	15	50	Desarrollo	13/11/17	15/11/17
	HU_05	5		Desarrollo	16/11/17	16/11/17
	HU_06	20		Desarrollo	17/11/17	22/11/17
	HT_08	10		Pruebas	23/11/17	24/11/17
5	HU_07	10	50	Desarrollo	27/11/17	28/11/17
	HU_08	25		Desarrollo	29/11/17	05/12/17
	HU_09	10		Desarrollo	06/12/17	07/12/17
	HT_09	5		Pruebas	08/12/17	08/12/17
6	HU_10	25	50	Desarrollo	11/12/17	15/12/17
	HU_11	20		Desarrollo	18/12/17	21/12/17
	HT_10	5		Pruebas	22/12/17	22/12/17
7	HU_12	15	50	Desarrollo	26/12/17	28/12/17
	HU_13	25		Desarrollo	29/12/17	05/01/18
	HT_11	10		Pruebas	08/01/18	09/01/18
8	HU_14	25	50	Desarrollo	10/01/18	16/01/18
	HU_15	15		Desarrollo	17/01/18	19/01/18
	HT_12	10		Pruebas	22/01/18	23/01/18
9	HU_16	5	50	Desarrollo	24/01/18	24/01/18
	HU_17	30		Desarrollo	25/01/18	01/02/18
	HT_13	15		Pruebas	02/02/18	06/02/18
10	HU_18	20	50	Desarrollo	07/02/18	14/02/18
	HT_05	30		Documentación	15/02/18	22/02/18
Puntos totales: 500				**Planificación: Del** 02/10/17 **Al** 22/02/18		

Realizado por: Wilmer B. 2018.

2.7 Análisis y selección de pruebas automáticas

Para la selección de los casos de prueba se ha tomado en cuenta solo las funcionalidades en sí ya sean estas metáforas o historias de usuario, las mismas se programarán para que estas funcionen de manera automática.

Cabe recalcar que al software se le aplicarán pruebas unitarias, más no pruebas de integración directamente, pruebas de regresión o pruebas de interfaces funcionales.

Dentro de cada uno de los sprints se ha ido seleccionando estas funcionalidades evitando la redundancia de procesos, y en cada funcionalidad o Historia de Usuario se ha analizado los casos de prueba que normalmente se haría con pruebas de aceptación siguiendo la metodología scrum.

2.7.1 Pruebas seleccionadas por historia de usuario

Tabla 10-2: Casos de prueba.

Historias de usuario	#	Casos de prueba
HU_01 Codificación de scripts SQL para transacciones en base de datos**. Sprint 2**	1	Verificar que la base de datos local haya sido creada
	2	Verificar eliminación de tabla COOPERATIVA
	3	Verificar eliminación de tabla DEPENDENCIA
	4	Verificar eliminación de tabla DEPENDENCIA POSEE RUTAS
	5	Verificar eliminación de tabla HORARIO
	6	Verificar eliminación de tabla RUTA
	7	Verificar eliminación de tabla TERMINAL
	8	Verificar eliminación de tabla TERMINAL PERTENCE DEPENDENCIA
	9	Verificar eliminación de tabla PROVINCIAS
	10	Verificar eliminación de tabla CIUDADES
	11	Verificar eliminación de tabla FAVORITO
	12	Verificar eliminación del contenido existente en tabla COOPERATIVA en base de datos local
	13	Verificar eliminación del contenido existente en tabla DEPENDENCIA en base de datos local
	14	Verificar eliminación del contenido existente en tabla DEPENDENCIA POSEE RUTAS en base de datos local
	15	Verificar eliminación del contenido existente en tabla HORARIO en base de datos local
	16	Verificar eliminación del contenido existente en tabla RUTA en base de datos local
	17	Verificar eliminación del contenido existente en tabla TERMINAL en base de datos local
	18	Verificar eliminación del contenido existente en tabla TERMINAL PERTENCE DEPENDENCIA en base de datos local
	19	Verificar eliminación del contenido existente en tabla PROVINCIAS en base de datos local
	20	Verificar eliminación del contenido existente en tabla CIUDADES en base de datos local
	21	Verificar eliminación del contenido existente en tabla FAVORITO en base de datos local
	22	Verificar la creación de tabla COOPERATIVA en base de datos local
	23	Verificar la creación de tabla DEPENDENCIA en base de datos local

	24	Verificar la creación de tabla DEPENDENCIA POSEE RUTAS en base de datos local
	25	Verificar la creación de tabla HORARIO en base de datos local
	26	Verificar la creación de tabla RUTA en base de datos local
	27	Verificar la creación de tabla TERMINAL en base de datos local
	28	Verificar la creación de tabla TERMINAL PERTENCE DEPENDENCIA en base de datos local
	29	Verificar la creación de tabla PROVINCIAS en base de datos local
	30	Verificar la creación de tabla CIUDADES en base de datos local
	31	Verificar la creación de tabla FAVORITO en base de datos local
HU_03 Implementación de servicios web Geocoding API de Google Maps a través de petición GET**. Sprint 3**	1	Verificar que haya respuesta del consumo de servicio web de Google Maps
	2	Chequear cuando el consumo de servicio web geocoding no fue exitoso
	3	Chequear cuando consumo de servicio web geocoding fue exitosa pero no obtuvo resultados
	4	Verificar consumo de servicio web geocoding sea exitoso
	5	Verificar que la respuesta del consumo de servicio web geocoding obtuvo una lista de resultados
	6	Verificar que la conexión http es aceptada
	7	Chequear que la conexión htttp fue rechazada
HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps**. Sprint 4**	1	Chequear que la lista de direcciones filtradas esté vacía
	2	Verificar que existe lista de direcciones filtradas
	3	Formatear lista de direcciones filtradas
	4	Cargar provincias desde base de datos local en una lista
	5	Chequear que la lista de provincias de base de datos local este vacía
	6	Encontrar coincidencia de provincia entre la lista de direcciones y la lista de provincias extraídas de la base de datos local
	7	Chequear cuando no se encuentra coincidencia de provincia entre la lista de direcciones y la lista de provincias extraídas de la base de datos local
	8	Cargar lista de ciudades desde base de datos local en una lista, en base al id de una provincia
	9	Chequear cuando la lista de ciudades extraídas de la base en base al id de una provincia sea vacía
	10	Encontrar coincidencia de ciudad entre la lista de direcciones y la lista de ciudades extraídas de la base de datos
	11	Chequear cuando no se encontraron coincidencias de ciudad entre la lista de direcciones y la lista de ciudades extraídas de la base de datos
	12	Obtener distancia entre dos coordenadas gps a través de geometría esférica
	13	Transformar cantidad matemática normal a radianes
	14	Obtener el índice del número menor de una lista de distancias entre ciudades
	15	Cargar todas las ciudades desde base de datos local en una lista
	16	Chequear cuando la lista de todas las ciudades extraída de la base de datos local esté vacía
	17	Verificar ciudad cercana en base a otra

	18	Chequear cuando no existe ciudad cercana en base a otra
	19	Chequear cuando si existe ciudad cercana en base a otra
	20	Verificar la existencia de una ciudad específica dentro de la base de datos en base a una ciudad ingresada
	21	Verificar si no existen ciudad origen, en base a una ciudad ingresada
HU_06 Implementación de funcionalidad para consulta de horarios por transporte**. Sprint 4**	1	Cargar lista de cooperativas de transporte desde la base de datos local
	2	Chequear cuando la lista de transportes esté vacía, cuando no se encuentre ninguna cooperativa de transporte
	3	Convertir lista de objetos de cooperativas de transporte a lista de strings
	4	Cargar lista de rutas en base al id de una cooperativa de transporte desde la base de datos local
	5	Chequear cuando la lista de rutas en base al id de una cooperativa de transportes esté vacía
	6	Extraer lista de rutas de lista de objetos dependencia a lista de strings ordenadas alfabéticamente
	7	Obtener id de dependencia posee ruta en base a ciudad origen y destino de una cooperativa de transporte
	8	No se ha obtenido id de dependencia posee ruta en base a ciudad origen y destino de una cooperativa de transporte
	9	Obtener las dependencias o agencia junto con las rutas, en base al id de la misma
	10	Verificar que no existe información de dependencia o agencia que poseen rutas, en base a su id
	11	Obtener lista de horarios de una ruta en base al id de Dependencia posee ruta
	12	Verificar que no existan datos registrados de los horarios de una ruta en base al id de dependencia posee ruta
HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino**. Sprint 5**	1	Cargar lista ciudades origen desde base de datos local
	2	Chequear cuando no hay lista de ciudades origen desde base de datos local vacía
	3	Cargar lista ciudades destino desde la base de datos local
	4	Chequear cuando la lista ciudades destino está vacía
	5	Extraer lista de Strings de ciudades destino de una lista de objetos de ruta
	6	Formatear cadena de texto a formato donde las palabras empiezan con letra mayúscula
	7	Obtener lista de horarios en base a una ruta específica
	8	Chequear que la lista de horarios en base a una ruta específica este vacía
	9	Obtener id de una ruta
	10	Chequear cuando no se ha obtenido id de ruta
HU_09 Implementación de funcionalidad para obtener la próxima salida en base a la hora actual**. Sprint 5**	1	Adaptar lista de información de horarios, rutas y transportes para presentación en interfaz
	2	Ordenar ascendente la lista de horarios de una ruta específica adaptada
	3	Verificar la hora actual es AM
	4	Verificar la hora actual es PM

	5	Obtener el día de la semana actual numéricamente
	6	Convertir día numérico actual a día actual escrito
	7	Obtener hora actual normal
HU_11 Implementación de funcionalidad para mostrar horarios por días**. Sprint 6**	1	Convertir día numérico a abreviación de día
	2	Concatenación de días numéricos a días escritos laborables
HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android**. Sprint 7**	1	Activar servicio para localización LocationManager para uso de servicio de localización
	2	Verificar proveedor de localización gps activo
	3	Verificar proveedor de localización gps inactivo
	4	Verificar proveedor de localización de red activo
	5	Verificar proveedor de localización de red inactivo
	6	Omitir dato altitud en la localización
	7	Activar consumo de energía bajo
	8	Existe proveedor de localización activo
	9	Chequear cuando no existe proveedor de localización activo
	10	Desactivar servicio de localización, cuando ya no se necesita
	11	Chequear que el mejor proveedor de localización no sea el de red
	12	Verificar que el proveedor de localización esté en modo pasivo
	13	Verificar que el mejor proveedor de localización sea el dispositivo gps
	14	Chequear que el mejor proveedor de localización no sea el dispositivo gps
	15	Verificar que el proveedor de localización gps esté inactivo
HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte**. Sprint 7**	1	Verificar si existe una conexión a internet activa
	2	Chequear que haya conexión activa de red wifi
	3	Chequear cuando no hay conexión activa de red wifi
	4	Verificar que haya conexión activa de datos móviles
	5	Chequear cuando no hay conexión activa de datos móviles
	6	Activar servicio para acceso a internet
	7	Chequear cuando no existe servicio activo para acceso a internet
	8	Verificar si existe conectividad y disponibilidad a datos móviles
	9	Verificar si fue exitoso el abrir una petición http
	10	Verificar estado de la petición sea aceptada
	11	Chequear cuando la petición http fue rechazada
	12	Verificar resultado de petición http respuesta no esté vacía
	13	Chequear cuando no hay resultado de consumo de servicio web
	14	Verificar la disponibilidad de asientos de la respuesta traída de la petición http
	15	Verificar que el “Resultado” a la consulta de boletos disponibles sea una consulta correcta “CC”
	16	Verificar que el “Resultado” a la consulta de boletos disponibles sea una consulta incorrecta “CI”
	17	Verificar el número de asientos disponibles
	18	Verificar la cantidad de asientos ocupados

HU_14 Implementación de funcionalidad favoritos**. Sprint 8**	1	Cargar lista de rutas favoritas extraída desde base de datos local
	2	Chuequear cuando no haya lista de favoritos extraída de la base de datos
	3	Convertir lista de objeto de favoritos a lista de cadena string de favoritos
	4	Agregar ruta a favorito
	5	Verificar si dicho favorito ya existe
	6	Obtener la ruta favorita a partir del id y la hora de la misma
	7	Verificar si una ruta específica está registrada dentro de favorito
HU_15 Implementación de funcionalidad llamada telefónica y cargar datos de agencia de cooperativa**. Sprint 8**	1	Obtener datos de la agencia
	2	Chequear cuando no existe datos de la agencia
	3	Verificar permiso de llamada activo
	4	Chequear cuando no existe permiso de llamada activo
	5	Convertir número string a formato de número para llamar
HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting**. Sprint 9**	1	Activar servicio de wifi
	2	Verificar conexión a la red activa
	3	Chequear cuando no hay conexión activa hacia la red
	4	Verificar que el servicio de conexión a la red esté disponible
	5	Verificar que el tipo de conexión sea wifi
	6	Verificar el tipo de conexión de red sea datos móviles
	7	Verificar que la respuesta del servicio web traiga un “Resultado” con “CC” de consulta correcta
	8	Verificar que la respuesta del servicio web traiga un “Resultado” con “CI” de consulta incorrecta
	9	Verificar que existan datos de la tabla “COOPERATIVA”
	10	Verificar que existan datos de la tabla “DEPENDENCIA”
	11	Verificar que existan datos de la tabla “TERMINAL”
	12	Verificar que existan datos de la tabla “RUTA”
	13	Verificar que existan datos de la tabla “HORARIO”
	14	Verificar que existan datos de la tabla “TERM_PERTENECE_DEPEN”
	15	Verificar que existan datos de la tabla “DEPEN_POSEE_RUTAS”
	16	Verificar que existan datos de la tabla “PROVINCIAS”
	17	Verificar que existan datos de la tabla “CIUDADES”
	18	Verificar que existan datos de la tabla “FAVORITO”
	19	Verificar eliminación de datos anteriores de la base de datos local
	20	Verificar inserción de datos de la tabla “COOPERATIVA” desde el servicio web en tablas locales de la base de datos
	21	Verificar inserción de datos de la tabla “DEPENDENCIA” desde el servicio web en tablas locales de la base de datos
	22	Verificar inserción de datos de la tabla “TERM_PERTENECE_DEPEN” desde el servicio web en tablas locales de la base de datos
	23	Verificar inserción de datos de la tabla “TERMINAL” desde el servicio web en tablas locales de la base de datos
	24	Verificar inserción de datos de la tabla “DEPEN_POSEE_RUTAS” desde el servicio web en tablas locales de la base de datos

	25	Verificar inserción de datos de la tabla “RUTA” desde el servicio web en tablas locales de la base de datos
	26	Verificar inserción de datos de la tabla “HORARIO” desde el servicio web en tablas locales de la base de datos
	27	Verificar inserción de datos de la tabla “CIUDAD” desde el servicio web en tablas locales de la base de datos
	28	Verificar inserción de datos de la tabla “PROVINCIA” desde el servicio web en tablas locales de la base de datos
8 Sprints involucrados	**Total, casos de prueba:** 163	

Realizado por: Wilmer B. 2018.

2.8 Diseño de pruebas automáticas

2.8.1 Diseño de casos de prueba

Para el diseño de cada caso de prueba se proponen los siguientes formatos con los cuales se llevará a cabo las pruebas automáticas, las mismas se organizarán por sprints, y para cada sprint se ha definido los siguientes formatos:

- **Tabla de requisitos de test:** en esta tabla de definirán los requisitos necesarios, antes o después de ejecutar cualquier test.

  Pre-Ejecución: Son todas aquellas **Variables** que necesitan ser declaradas, también funciones que se deben ejecutar antes de llevar a cabo cualquier test automatizado.

  Métodos colaboradores: son aquellos métodos que se ejecutarán al momento de ejecutar una prueba unitaria, estos puedes ser vaciar una tabla, llenar una tabla en la base de datos, etc.

  Post Ejecución: Son aquellos métodos que se ejecutarán luego de haberse llevado a cabo todas las pruebas unitarias de un sprint, estos podrían ser eliminar objetos creados, cerrar conexiones a base de datos, etc.

Tabla 11-2: Ejemplo modelo requisitos de test.

REQUISITOS DE TEST SPRINT 4, HU_01

Pre-Ejecución

Variables:

```
private Context contexto;
private BaseDatos baseDatos;
private ArrayList<Provincia> lstProvincias;
private ArrayList<Ciudad> lstCiudades;
```

Función:

```
@Before
public void setUp() throws Exception {
   contexto=InstrumentationRegistry.getTargetContext();
   baseDatos=new BaseDatos(contexto);
}
```

Métodos Colaboradores

```
public void llenarProvincias()
{
   lstProvincias= new ArrayList<>();
   lstProvincias.add(new Provincia(1,"azuay"));
   lstProvincias.add(new Provincia(2,"bolívar"));
   lstProvincias.add(new Provincia(3,"cañar"));
   lstProvincias.add(new Provincia(4,"carchi"));
   lstProvincias.add(new Provincia(5,"chimborazo"));
}

public void llenarCiudades()
{
   lstCiudades= new ArrayList<>();
   lstCiudades.add(new Ciudad("riobamba",-1.6732642,-78.6581558));
   lstCiudades.add(new Ciudad("quito",-0.1568747,-78.540189));
   lstCiudades.add(new Ciudad("cuenca",-2.8987054,-78.996889));
   lstCiudades.add(new Ciudad("puyo",-1.4892927,-78.009359));
}
```

Post Ejecución

Función:

```
@After
public void tearDown() throws Exception {
   baseDatos.close();
}
```

Realizado por: Wilmer B. 2018.

- **Caso de prueba automatizada:** la siguiente tabla corresponde al conjunto de pruebas seleccionadas dentro de cada historia de técnica o de usuario, las mismas que se nombrarán

como casos de prueba y funcionarán de una manera automática. Dentro de los campos que definen este formato tenemos los siguientes:

Sprint No: Es el número de sprint en el cual se encuentra dicha prueba.

Funcionalidad: Corresponde a la historia de usuario en la cual se encuentra dicha prueba, la misma debe iniciar primero con el id de la historia de usuario, seguido por el nombre de esta.

Código: Es el identificador único con el cual se localiza dicho caso de prueba, este identificador comenzará con la abreviación TC (Test Case) separado por un guion bajo de un número secuencial el cual se volverá a iniciar en 01 en cada nuevo sprint. Ejemplo TC_01.

Descripción: Es una breve descripción de qué es lo que se va a hacer en tal caso de prueba.

Entradas: En sí corresponde al código fuente usado en tal caso de prueba, juntos con los datos de entrada que se utilizan en la misma.

Resultado esperado: Corresponde a como se espera que reaccione la funcionalidad ante la prueba aplicada, se detalla de manera específica cual debería ser el resultado en caso de ser exitosa.

Evaluación de la prueba: Esta podrá tener dos estados los cuales definen a la prueba como exitosa o fallida.

Tiempo de ejecución: Es el tiempo correspondiente que tomo realizar dicho caso de prueba.

Tabla 12-2: Ejemplo modelo caso de prueba automatizada.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_01 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_01	**Descripción:** Chequear que la lista de direcciones filtradas esté vacía
Entrada(s)	

```
@Test
public void listaDireccionesVacia()
{
   JSONArray arrayJson= null;
   ArrayList<String> respuesta=null;
```

```
    try {
        arrayJson = new JSONArray("[]");
        Geolocalizacion geo=new Geolocalizacion();
        respuesta=geo.filtrarDatosSWgoogleMaps(arrayJson);
    } catch (JSONException e) {
        e.printStackTrace();
    }
    assertEquals(respuesta,null);
}
```

Resultado Esperado	
Cuando la lista de direcciones tiene un formato json incorrecto, entonces en el filtrado de estas devolverá un valor **null**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

- **Resultado de pruebas por sprint:** dicha tabla contiene un breve resumen de la aplicación de casos de pruebas automatizadas por sprint. La cual consta de los siguientes campos:

  Cantidad de pruebas realizadas: Es el número total de pruebas aplicados en dicho sprint.

  Tiempo total en ejecución de prueba Sprint: es la suma de todos los tiempos que se ha obtenido en cada uno de los casos de prueba automatizados, después de ser ejecutados.

Tabla 13-2: Ejemplo modelo resultado de pruebas por historia de usuario en un sprint.

RESULTADO	
Cantidad de pruebas	20
Tiempo total en ejecución de pruebas Sprint 4, HU_01	5s 408ms

Realizado por: Wilmer B. 2018.

Para llevar a cabo la medición de tiempos que se demora probar las funcionalidades, tanto de manera automática como de manera manual (personal humano) se ha tomado en cuenta una un total de 153 casos de prueba correspondientes a 12 Historias de usuario funcionales, las mismas que se encuentran distribuidas en 8 Sprints.

2.8.2 Estructuración de pruebas automáticas en la aplicación

Para proceder a la implementación de los casos de prueba junto con el desarrollo de la aplicación, antes se presentará un diagrama del flujo que seguirán las distintas pruebas dentro de la aplicación móvil, mostrando así o dando una idea de cómo actúan estas pruebas automatizadas sobre el sistema.

También este flujo muestra una jerarquía piramidal en la que la suite forma o gestiona caminos de evaluación dentro del sistema, existiendo 3 caminos principales por los que se puede guiar, en el primer y segundo camino se encuentran las pruebas instrumentadas, mismas que utilizan servicios que deben ser permitidos o rechazados por el dispositivo, mientras que el otro camino son pruebas de unidad local, estas pueden ser pruebas con el fin de realizar cálculos, procesar información, realizar transacciones a la base de datos ya que no se necesita de un dispositivo ni del emulador virtual sino simplemente se puede utilizar la máquina virtual de java.

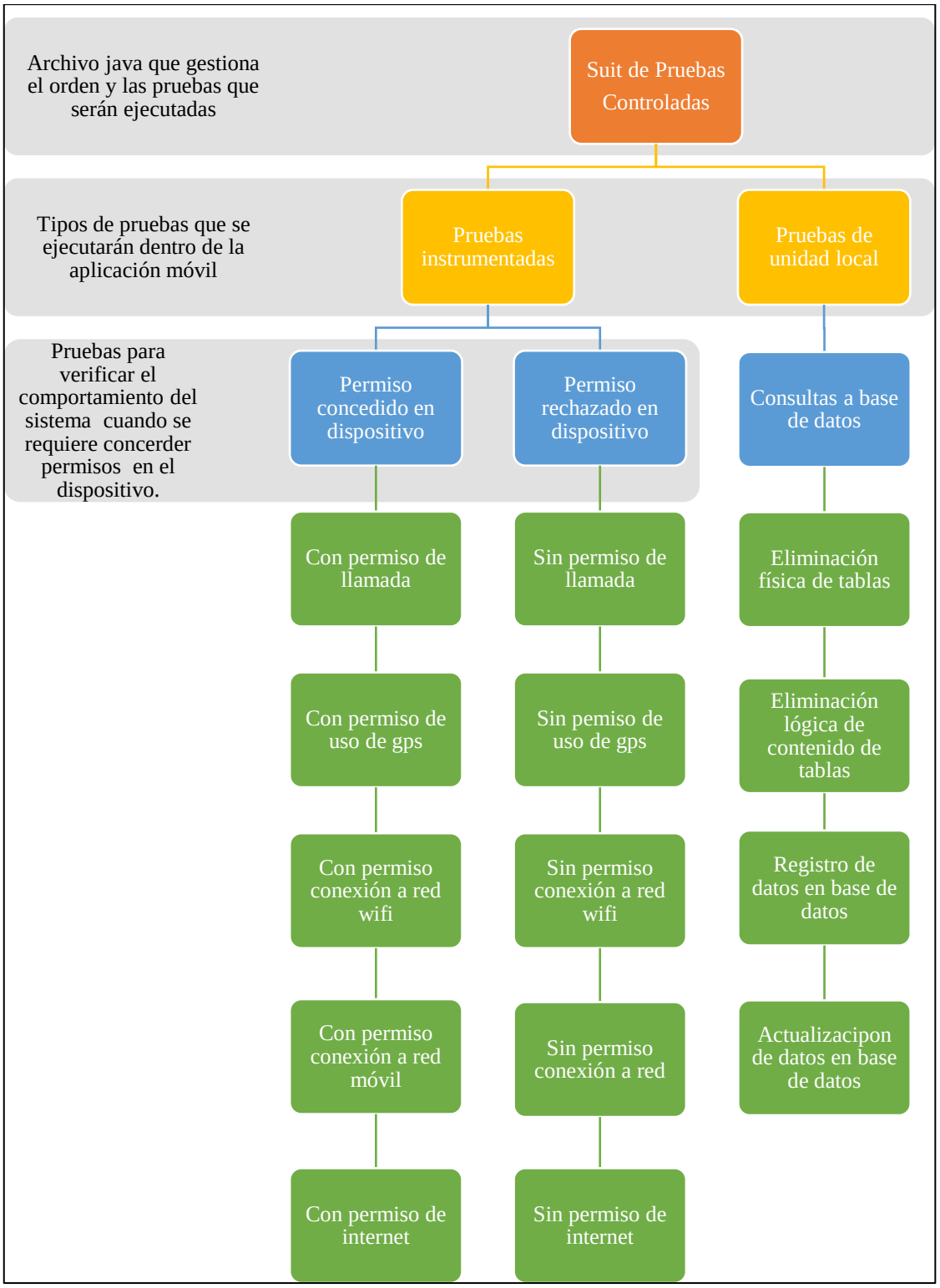


Gráfico 1-2: Diseño de casos de prueba.

Realizado por: Wilmer B. 2018.

2.9 Diseño de estándar de codificación

Para el desarrollo del presente proyecto se seguirá el estándar de codificación UpperCamelCase definido en la convención de código java 1995-1999. Este consiste básicamente en que cada nueva palabra tiene que iniciar con una letra mayúscula excepto la primera letra de la primera palabra. También se suele adaptar este estándar de acuerdo a las necesidades de desarrolladores dentro de una empresa o grupo de trabajo, teniendo así variaciones de este. En este proyecto se harán ciertas adaptaciones por comodidad personal a la hora de codificar.

Organización de ficheros

Un fichero consiste de secciones que deben estar separadas por líneas en blanco y comentarios opcionales que identifican cada sección.

Ficheros fuente java

Cada fichero fuente Java contiene una única clase o interface pública. Cuando algunas clases o interfaces privadas están asociadas a una clase pública, pueden ponerse en el mismo fichero que la clase pública. La clase o interfaz pública debe ser la primera clase o interface del fichero.

Los ficheros fuentes Java tienen la siguiente ordenación:

- Comentarios de comienzo
- Sentencias package e import
- Declaraciones de clases e interfaces

Comentarios de comienzo

Todos los ficheros fuente deben comenzar con un comentario en el que se lista el nombre de la clase, información de la versión, fecha, y copyright:

```
/*
* Nombre de la clase
*
* Información de la versión
*
* Fecha
*
* Copyright
*/
```

Sentencias package e import

La primera línea no-comentario de los ficheros fuente Java es la sentencia package. Después de esta, pueden seguir varias sentencias import. Por ejemplo:

```
package java.awt;
import java.awt.peer.CanvasPeer;
```

Nota: El primer componente del nombre de un paquete único se escribe siempre en minúsculas con caracteres ASCII y debe ser uno de los nombres de dominio de último nivel, actualmente com, edu, gob, mil, net, org, o uno de los códigos ingleses de dos letras que especifican el país como se define en el ISO Standard 3166, 1981.

Declaraciones de clases e interfaces

La siguiente tabla describe las partes de la declaración de una clase o interface, en el orden en que deberían aparecer.

Tabla 14-2: Declaración de clases e interfaces en estándar de programación.

Orden	Declaración de clase o interface	Notas
1	Comentario de documentación de la clase o interface (/**...*/)	
2	Sentencia class o interface	
3	Comentario de implementación de la clase o interface si fuera necesario (/*...*/)	Este comentario debe contener cualquier información aplicable a toda la clase o interface que no era apropiada para estar en los comentarios de documentación de la clase o interface.
4	Variables de clase (static)	Primero las variables de clase public, después las protected, después las de nivel de paquete (sin modificador de acceso), y después las private.
5	Variables de instancia	Primero las public, después las protected, después las de nivel de paquete (sin modificador de acceso), y después las private
6	Constructores	
7	Métodos	Estos métodos se deben agrupar por funcionalidad más que por visión o accesibilidad. Por ejemplo, un método de clase privado puede estar entre dos métodos públicos de instancia. El objetivo es hacer el código más legible y comprensible.

Realizado por: javaHispano. 2001.

Indentación

Se deben emplear cuatro espacios como unidad de indentación. La construcción exacta de la indentación (espacios en blanco contra tabuladores) no se especifica. Los tabuladores deben ser exactamente cada 8 espacios (no 4).

Longitud de la línea

Evitar las líneas de más de 80 caracteres, ya que no son manejadas bien por muchas terminales y herramientas.

Nota: Ejemplos para uso en la documentación deben tener una longitud inferior, generalmente no más de 70 caracteres.

Rompiendo líneas

Cuando una expresión no entre en una línea, romperla de acuerdo con estos principios:

- Romper después de una coma.
- Romper antes de un operador.
- Preferir roturas de alto nivel (más a la derecha que el "padre") que de bajo nivel (más a la izquierda que el "padre").
- Alinear la nueva línea con el comienzo de la expresión al mismo nivel de la línea anterior.
- Si las reglas anteriores llevan a código confuso o a código que se aglutina en el margen derecho, indentar justo 8 espacios en su lugar.

Ejemplos de cómo romper la llamada a un método:

```
unMetodo(expresionLarga1, expresionLarga2, expresionLarga3,
 expresionLarga4, expresionLarga5);

var = unMetodo1(expresionLarga1,
 unMetodo2(expresionLarga2,
 expresionLarga3));
```

Ahora dos ejemplos de ruptura de líneas en expresiones aritméticas. Se prefiere el primero, ya que el salto de línea ocurre fuera de la expresión que encierra los paréntesis.

```
nombreLargo1 = nombreLargo2 * (nombreLargo3 + nombreLargo4
 - nombreLargo5) + 4 * nombreLargo6; // PREFERIDA
nombreLargo1 = nombreLargo2 * (nombreLargo3 + nombreLargo4
```

```
 - nombreLargo) + 4 * nombreLargo6;
// EVITAR
```

Ahora dos ejemplos de indentación de declaraciones de métodos. El primero es el caso convencional. El segundo conduciría la segunda y la tercera línea demasiado hacia la izquierda con la indentación convencional, así que en su lugar se usan 8 espacios de indentación.

```
//INDENTACION CONVENCIONAL
unMetodo(int anArg, Object anotherArg, String yetAnotherArg,
        Object andStillAnother) {
    ...
}

//INDENTACION DE 8 ESPACIOS PARA EVITAR GRANDES INDENTACIONES
private static synchronized metodoDeNombreMuyLargo(int unArg,
        Object otroArg, String todaviaOtroArg,
        Object unOtroMas) {
    ...
}
```

Saltar de líneas por sentencias if deberá seguir generalmente la regla de los 8 espacios, ya que la indentación convencional (4 espacios) hace difícil ver el cuerpo. Por ejemplo:

```
//NO USAR ESTA INDENTACION
if ((condicion1 && condicion2)
    || (condicion3 && condicion4)
    ||!(condicion5 && condicion6)) { //MALOS SALTOS
    hacerAlgo();                    //HACEN ESTA LINEA FACIL DE OLVIDAR
}

//USAR ESTA INDENRACIÓN EN SU LUGAR
if ((condicion1 && condicion2)
        || (condicion3 && condicion4)
        ||!(condicion5 && condicion6)) {
    hacerAlgo();
}
//O USAR ESTA
```

```
if ((condicion1 && condicion2) || (condicion3 && condicion4)
    ||!(condicion5 && condicion6)) {
  hacerAlgo();
}
```

Hay tres formas aceptables de formatear expresiones ternarias:

```
alpha = (unaLargaExpresionBooleana) ? beta : gamma;

alpha = (unaLargaExpresionBooleana) ? beta
                                    : gamma;
alpha = (unaLargaExpresionBooleana)
        ? beta
        : gamma;
```

Comentarios

Los programas Java pueden tener dos tipos de comentarios: comentarios de implementación y comentarios de documentación. Los comentarios de implementación son aquellos que también se encuentran en C++, delimitados por /*...*/, y //.

Comentarios de bloque

Los comentarios de bloque se usan para dar descripciones de ficheros, métodos, estructuras de datos y algoritmos. Los comentarios de bloque se podrán usar al comienzo de cada fichero o antes de cada método. También se pueden usar en otros lugares, tales como el interior de los métodos. Los comentarios de bloque en el interior de una función o método deben ser indentados al mismo nivel que el código que describen.

Un cometario de bloque debe ir precedido por una línea en blanco que lo separe del resto del código.

```
/*
* Aquí hay un comentario de bloque.
*/
```

Los comentarios de bloque pueden comenzar con /*-, que es reconocido por indent(1) como el comienzo de un comentario de bloque que no debe ser reformateado. Ejemplo:

```
/*-
* Aquí tenemos un comentario de bloque con cierto
```

```
* formato especial que quiero que ignore indent(1).
*
*          uno
*                    dos
*                              tres
*/
```

Comentarios de una línea

Pueden aparecer comentarios cortos de una única línea al nivel del código que siguen. Si un comentario no se puede escribir en una línea, debe seguir el formato de los comentarios de bloque. Un comentario de una sola línea debe ir precedido de una línea en blanco. Aquí un ejemplo de comentario de una sola línea en código Java:

```
if (condición) {
   /* Código de la condición. */
   ...
}
```

Comentarios de remolque

Pueden aparecer comentarios muy pequeños en la misma línea que describen, pero deben ser movidos lo suficientemente lejos para separarlos de las sentencias. Si más de un comentario corto aparecen en el mismo trozo de código, deben ser indentados con la misma profundidad.
Aquí un ejemplo de comentario de remolque:

```
if (a == 2) {
   return TRUE; /* caso especial */
} else {
   return isPrime(a); /* caso gerenal */
}
```

Comentarios de fin de línea

El delimitador de comentario // puede convertir en comentario una línea completa o una parte de una línea. No debe ser usado para hacer comentarios de varias líneas consecutivas; sin embargo, puede usarse en líneas consecutivas para comentar secciones de código.
Aquí hay ejemplos de los tres estilos:

```
if (foo > 1) {
```

```
    // Hacer algo.
    ...
}
else {
    return false; // Explicar aquí por qué.
}
//if (bar > 1) {
//
//    // Hacer algo.
//     ...
//}
//else {
//    return false;
//}
```

Declaraciones

Cantidad por línea

Se recomienda una declaración por línea, ya que facilita los comentarios. En otras palabras, se prefiere.

```
int nivel; // nivel de indentación
int tam; // tamaño de la tabla
```

antes que

```
int level, size;
```

No poner diferentes tipos en la misma línea. Ejemplo:

```
int foo, fooarray[]; //ERROR!
```

Nota: Los ejemplos anteriores usan un espacio entre el tipo y el identificador. Una alternativa aceptable es usar tabuladores, por ejemplo:

```
int level; // nivel de indentación
int size; // tamaño de la tabla
Object currentEntry; // entrada de la tabla seleccionada actualmente
```

Inicialización

Intentar inicializar las variables locales donde se declaran. La única razón para no inicializar una variable donde se declara es si el valor inicial depende de algunos cálculos que deben ocurrir.

Colocación

Poner las declaraciones solo al principio de los bloques (un bloque es cualquier código encerrado por llaves "{" y "}".) No esperar al primer uso para declararlas; puede confundir a programadores no preavisados y limitar la portabilidad del código dentro de su ámbito de visibilidad.

```
void myMethod() {
   int int1 = 0; // comienzo del bloque del método
   if (condición) {
      int int2 = 0; // comienzo del bloque del "if"
      ...
   }
}
```

La excepción de la regla es los índices de bucles for, que en Java se pueden declarar en la sentencia for:

```
for (int i = 0; i < maximoVueltas; i++) { ... }
```

Evitar las declaraciones locales que ocultan declaraciones de niveles superiores. por ejemplo, no declarar la misma variable en un bloque interno:

```
int cuenta;
...
miMetodo() {
   if (condición) {
      int cuenta = 0; // EVITAR!
      ...
   }
   ...
}
```

Declaraciones de class e interfaces

Al codificar clases e interfaces de Java, se siguen las siguientes reglas de formato:

- Ningún espacio en blanco entre el nombre de un método y el paréntesis "(" que abre su lista de parámetros
- La llave de apertura "{" aparece al final de la misma línea de la sentencia declaración
- La llave de cierre "}" empieza una nueva línea indentada para ajustarse a su sentencia de apertura correspondiente, excepto cuando no existen sentencias entre ambas, que debe aparecer inmediatamente después de la de apertura "{"

```
class Ejemplo extends Object {
    int ivar1;
    int ivar2;

    Ejemplo(int i, int j) {
        ivar1 = i;
        ivar2 = j;
    }

    int metodoVacio() {}

    ...
}
```

- Los métodos se separan con una línea en blanco

Sentencias

Sentencias simples

Cada línea debe contener como mucho una sentencia. Ejemplo:

```
argv++; // Correcto
argc--; // Correcto
argv++; argc--; // EVITAR!
```

Sentencias compuestas

Las sentencias compuestas son sentencias que contienen listas de sentencias encerradas entre llaves “{sentencias}". Ver las siguientes secciones para ejemplos.

- Las sentencias encerradas deben indentarse un nivel más que la sentencia compuesta.

- La llave de apertura se debe poner al final de la línea que comienza la sentencia compuesta; la llave de cierre debe empezar una nueva línea y ser indentada al mismo nivel que el principio de la sentencia compuesta.
- Las llaves se usan en todas las sentencias, incluso las simples, cuando forman parte de una estructura de control, como en las sentencias if-else o for. Esto hace más sencillo añadir sentencias sin incluir bugs accidentalmente por olvidar las llaves.

Sentencias de retorno

Una sentencia return con un valor no debe usar paréntesis a menos que hagan el valor de retorno más obvio de alguna manera. Ejemplo:

```
return;
return miDiscoDuro.size();
return (tamanyo ? tamanyo : tamanyoPorDefecto);
```

Sentencias if, if-else, if else-if else

La clase de sentencias if-else debe tener la siguiente forma:

```
if (condición) {
   sentencias;
}

if (condición) {
   sentencias;
} else {
   sentencias;
}

if (condición) {
   sentencia;
} else if (condición) {
   sentencia;
} else {
   sentencia;
}
```

Nota: Las sentencias if usan siempre llaves {}. Evitar la siguiente forma, propensa a errores:

```
if (condición) //EVITAR! ¡ESTO OMITE LAS LLAVES {}!
    sentencia;
```

Sentencias for

Una sentencia for debe tener la siguiente forma:

```
for (inicialización; condición; actualización) {
   sentencias;
}
```

Una sentencia for vacía (una en la que todo el trabajo se hace en las cláusulas de inicialización, condición, y actualización) debe tener la siguiente forma:

```
for (inicialización; condición; actualización);
```

Al usar el operador coma en la cláusula de inicialización o actualización de una sentencia for, evitar la complejidad de usar más de tres variables. Si se necesita, usar sentencias separadas antes de bucle for (para la cláusula de inicialización) o al final del bucle (para la cláusula de actualización).

Sentencias while

Una sentencia while debe tener la siguiente forma:

```
while (condición) {
   sentencias;
}
```

Una sentencia while vacía debe tener la siguiente forma:

```
while (condición);
```

Sentencias do-while

Una sentencia do-while debe tener la siguiente forma:

```
do {
   sentencias;
} while (condición);
```

Sentencias switch

Una sentencia switch debe tener la siguiente forma:

```
switch (condición) {
case ABC:
     sentencias;
     /* este caso se propaga */
    break;
case DEF:
    sentencias;
    break;

case XYZ:
    sentencias;
    break;

default:
    sentencias;
    break;
}
```

Cada vez que un caso se propaga (no incluye la sentencia break), añadir un comentario donde la sentencia break se encontraría normalmente. Esto se muestra en el ejemplo anterior con el comentario /* este caso se propaga */.

Cada sentencia switch debe incluir un caso por defecto. El break en el caso por defecto es redundante, pero prevé que se propague por error si luego se añade otro caso.

Sentencias try-catch

Una sentencia try-catch debe tener la siguiente forma:

```
try {
    sentencias;
} catch (ExceptionClass e) {
    sentencias;
}
```

Una sentencia try-catch puede ir seguida de un finally, cuya ejecución se ejecutará independientemente de que el bloque try se haya completado con éxito o no.

```
try {
   sentencias;
} catch (ExceptionClass e) {
   sentencias;
} finally {
   sentencias;
}
```

Espacio en blanco

Líneas en blanco

Las líneas en blanco mejoran la facilidad de lectura separando secciones de código que están lógicamente relacionadas.

Se deben usar siempre dos líneas en blanco en las siguientes circunstancias:

- Entre las secciones de un fichero fuente
- Entre las definiciones de clases e interfaces.

Se debe usar siempre una línea en blanco en las siguientes circunstancias:

- Entre métodos
- Entre las variables locales de un método y su primera sentencia
- Antes de un comentario de bloque o de un comentario de una línea
- Entre las distintas secciones lógicas de un método para facilitar la lectura.

Espacios en blanco

Se deben usar espacios en blanco en las siguientes circunstancias:

- Una palabra clave del lenguaje seguida por un paréntesis debe separarse por un espacio. Ejemplo:

```
while (true) {
    ...
}
```

Notar que no se debe usar un espacio en blanco entre el nombre de un método y su paréntesis de apertura. Esto ayuda a distinguir palabras claves de llamadas a métodos.

- Debe aparecer un espacio en blanco después de cada coma en las listas de argumentos.
- Todos los operadores binarios excepto. se deben separar de sus operandos con espacios en blanco. Los espacios en blanco no deben separador los operadores unarios, incremento ("++") y decremento ("--") de sus operandos. Ejemplo:

```
a += c + d;
a = (a + b) / (c * d);
while (d++ == s++) {
   n++;
}
printSize("el tamaño es " + foo + "\n");
```

- Las expresiones en una sentencia for se deben separar con espacios en blanco. Ejemplo:

```
for (expr1; expr2; expr3)
```

- Los "Cast"s deben ir seguidos de un espacio en blanco. Ejemplos:

```
miMetodo((byte) unNumero, (Object) x);
miMetodo((int) (cp + 5), ((int) (i + 3))
                                    + 1);
```

Convenciones de nombres

Las convenciones de nombres hacen los programas más entendibles haciéndolos más fácil de leer. También pueden dar información sobre la función de un identificador, por ejemplo, cuando es una constante, un paquete, o una clase, que puede ser útil para entender el código.

Tabla 15-2: Convenciones de nombre en estándar de codificación.

Tipos de identificadores	Reglas para nombres	Ejemplos
Paquetes	El prefijo del nombre de un paquete se escribe siempre con letras ASCII en minúsculas, y debe ser uno de los nombres de dominio de alto nivel, actualmente com, edu, gob, mil, net, org, o uno de los códigos ingleses de dos letras que identifican cada país como se especifica en el ISO Standard 3166, 1981. Los subsecuentes componentes del nombre del paquete variarán de acuerdo a las convenciones de nombres internas de cada organización. Dichas convenciones pueden especificar que algunos nombres de los directorios correspondan a divisiones, departamentos, proyectos o máquinas.	com.sun.eng com.apple.quicktime.v2 edu.cmu.cs.bovik.cheese
Clases	Los nombres de las clases deben ser sustantivos, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en mayúsculas. Intentar mantener los nombres de las clases simples y descriptivos. Usar palabras completas, evitar acrónimos y abreviaturas (a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL o HTML).	class Cliente; class ImagenAnimada;
Interfaces	Los nombres de las interfaces siguen la misma regla que las clases	interface ObjetoPersistente; interface Almacen;

Métodos	Los métodos deben ser verbos, cuando son compuestos tendrán la primera letra en minúscula, y la primera letra de las siguientes palabras que lo forma en mayúscula.	ejecutar();
		ejecutarRapido();
		cogerFondo();

Variables	Excepto las constantes, todas las instancias y variables de clase o método empezarán con minúscula. Las palabras internas que lo forman (si son compuestas) empiezan con su primera letra en mayúsculas. Los nombres de variables no deben empezar con los caracteres sub guion "_" o signo del dólar "$", aunque ambos están permitidos por el lenguaje. Los nombres de las variables deben ser cortos, pero con significado. La elección del nombre de una variable debe ser un mnemónico, designado para indicar a un observador casual su función. Los nombres de variables de un solo carácter se deben evitar, excepto para variables índices temporales. Nombres comunes para variables temporales son i, j, k, m, y n para enteros; c, d, y e para caracteres.	int i; char c; float miAnchura;

Constantes	Los nombres de las variables declaradas como constantes deben ir totalmente en mayúsculas separando las palabras con un sub guion ("_"). (Las constantes ANSI se deben evitar, para facilitar su depuración.)	static final int ANCHURA_MINIMA = 4;
		
		static final int ANCHURA_MAXIMA = 999;
		
		static final int COGER_LA_CPU = 1;

Realizado por: javaHispano. 2001.

Adapted with permission from JAVA CODE CONVENTIONS. Copyright 1995-1999 Sun Microsystems, Inc. All rights reserved.

Copyright 2001 www.javaHispano.com. Todos los derechos reservados. Otorgados derechos de copia y redistribución para uso interno y no comercial.

2.10 Fase de desarrollo de sprints

En la presenta fase se realizará el desarrollo de la app móvil y documentación de estas. El número de sprints totales son 10 los mismos que se desarrollará de acuerdo a la planificación establecida anteriormente. En esta sección del documento solo se mostrará un resumen de las actividades de cada sprint como también el desarrollo de las historias técnicas que forman parte de este proyecto. La documentación detallada se encuentra en el **Anexo B.**

2.10.1 Sprint 1

Tabla 16-2: Tabla de actividades del sprint 01.

ACTIVIDADES SPRINT 01			
Id	**Historia técnica/de usuario**	**Fecha**	**Estimación**
HT_01	Análisis de herramientas a emplear	02/10/17 – 03/10/17	10
HT_02	Análisis y gestión de riesgos	04/10/17 - 06/10/17	15
HT_03	Diseño de base de datos	09/10/17 - 13/10/17	25
		02/10/17 - 13/10/17	50

Realizado por: Wilmer B. 2018.

Para el presente sprint se ha realizado un total de 3 historias técnicas, 6 tareas de ingeniería y 6 pruebas manuales, con un total de 50 puntos. Llevándose a cabo desde el 02/10/17 hasta 13/10/17

siguiendo la planificación antes definida. La documentación completa que sustenta el desarrollo de las actividades se encuentra en el **Anexo B**.

2.10.1.1 HT_01 Análisis de herramientas a emplear

Tabla 17-2: Tabla historia técnica 01.

HISTORIA TÉCNICA		
Id: HT_01	**Nombre:** Análisis de herramientas a emplear	
Descripción: Como desarrollador necesito analizar el tipo software que se empleará en el desarrollo de este proyecto. Para así completarlo de una manera óptima.		
Usuario: Wilmer Barrera	**Sprint:** 1	
Fecha inicio: 02/10/17	**Fecha fin:** 03/10/17	**Esfuerzo:** 10

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Selección de herramientas a emplear.

Realizado por: Wilmer B. 2018.

Tabla 18-2: Tabla tarea de ingeniería 01 de historia técnica 01.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** análisis	**Sprint:** 1
Nombre de historia técnica: HT_01 Análisis de herramientas a emplear		
Nombre tarea: Selección de herramientas a emplear.		
Fecha inicio: 02/10/17		**Fecha fin:**03/10/17
Descripción: Como desarrollador necesito seleccionar las herramientas con las que se trabajará en el desarrollo del presente proyecto		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_01	Verificar que las herramientas seleccionadas presten las comodidades necesarias para llevar el proyecto a cabo de manera correcta.

Realizado por: Wilmer B. 2018.

A continuación, se muestra un resumen de tareas de ingeniería, así como pruebas de aceptación y/o pruebas automatizadas correspondientes al presente sprint. En el **Anexo B** se encuentra la documentación detallada del presente sprint.

Para poder desarrollar el proyecto, se ha optado por elegir las siguientes herramientas software con base en los conocimientos adquiridos a lo largo de la carrera y las ventajas que estos prestan para facilitar el trabajo en el desarrollo y gestión del presente proyecto.

Android Studio 2.2.3: Es el entorno de desarrollo integrado oficial de la compañía Google Inc. Tiene muchas ventajas, como instalar y crear propias máquinas virtuales, las cuales servirán como emuladores reemplazando dispositivos físicos, instalación de plugins para facilitar el trabajo en exportación de imágenes e íconos, predicción de escritura al momento de codificar, manipulación de emuladores y dispositivos desde modo consola, etc.

Sublime Text 3: Esta herramienta es un editor de código disponible para varios lenguajes de programación el mismo que presentas las siguientes ventajas: software gratuito, instalación de plugins para crear atajos a la hora de escribir código, software liviano, etc.

MySQL Workbench 6.3 CE: Esta herramienta se utilizará para realizar modelos lógicos de la base de datos, así como la obtención de scripts para realizar backups y gestionar la base de datos directamente.

SQLite 1: Es el gestor de base de datos por defecto que viene en los sistemas Android, su uso es bastante común gracias a la rapidez a la hora de procesar consultas y transacciones.

StarUML 2.7: Esta herramienta de modelado, se usará para realizar respectivos diagramas que ayudan a mostrar la estructura del proyecto, tales como: diagramas de clase, componentes, despliegue, casos de uso, etc.

Wamp Server 3.1.0: Esta herramienta se utilizará como servidor web local, ya que este software permite la configuración rápida y fácil de varias herramientas con las que se trabaja, tales como **PHP** y el gestor de base de datos más común como lo es **MySQL**. Entre las ventajas están: es una herramienta gratuita, configuración rápida, gestiona el uso de otras herramientas.

GanttProject: Esta herramienta se utilizará para la gestión del proyecto, planificaciones de entrega de resultados entre otros.

Tabla 19-2: Tabla prueba de aceptación 01.

PRUEBA DE ACEPTACIÓN	
Id: PA_01	**Nombre:** Verificar que las herramientas seleccionadas presten las comodidades necesarias para llevar el proyecto a cabo de manera correcta.
Tarea de ingeniería: TI_01 Selección de herramientas a emplear.	
Descripción: Para poder trabajar con las herramientas seleccionadas se debe tener un mínimo conocimiento sobre estas, o debe ser fácil de usar para que así el tiempo de aprendizaje empleado para aprender estos sea mínimo.	
Responsable: Wilmer Barrera	**Fecha:** 03/10/17
Condición de ejecución: - Tener herramientas software instaladas	
Pasos de ejecución: - Abrir cada una de las herramientas a usar - Configurar el ambiente de trabajo en cada herramienta - Verifica que cada herramienta funciona correctamente	
Resultado esperado: Las herramientas están configuradas y son óptimas para el desarrollo del proyecto	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

2.10.1.2 HT_02 Análisis y gestión de riesgos

Tabla 20-2: Tabla historia técnica 02.

HISTORIA TÉCNICA		
Id: HT_02	**Nombre:** Análisis y gestión de riesgos	
Descripción: Dentro del proyecto es importante ver los pros y contras que se pueden suscitar. Para esto de definirá la documentación de gestión y análisis de riesgos en los cuales se planificarán medidas a tomar en caso de que uno de los riesgos ocurriese.		
Usuario: Wilmer Barrera	**Sprint:** 1	
Fecha inicio: 04/10/17	**Fecha fin:** 06/10/17	**Esfuerzo:** 15
TAREAS DE INGENIERÍA		
Id	**Nombre**	
TI_01	Identificación y probabilidad de riesgos.	

Realizado por: Wilmer B. 2018.

Tabla 21-2: Tabla tarea de ingeniería 01 de historia técnica 02.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** análisis	**Sprint:** 1
Nombre de historia técnica: HT_02 Análisis y gestión de riesgos.		
Nombre tarea: Identificación y probabilidad de riesgos.		
Fecha inicio: 04/10/17		**Fecha fin:**06/10/17
Descripción: Como desarrollador necesito analizar los posibles riesgos que se pueden dar en el proceso de desarrollo del proyecto		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_02	Verificar que las posibles amenazas o riesgos en el proyecto tengan un plan de contingencia.	

Realizado por: Wilmer B. 2018.

Para llevar a cabo el análisis y la gestión de los riesgos, se ha optado por hacer un análisis de estos presentado en las siguientes tablas.

Tabla 22-2: Identificación de Riesgos.

ID	DESCRIPCIÓN	TIPO	CONSECUENCIA
R1	No se analizó correctamente los requerimientos	DEL PROYECTO	Retraso en las actividades del proyecto debido a los cambios de la planificación
R2	Desconocimiento de las herramientas que se van a utilizar	TÉCNICO	Retraso en la ejecución del proyecto
R3	Cambios de requerimientos por parte del cliente	DEL PROYECTO	Retraso en el proyecto.
R4	Daños en el equipo hardware de desarrollo	TÉCNICO	Retraso en la ejecución del proyecto.
R5	Falta de comunicación con el cliente	DEL PROYECTO	Retraso en la planificación.
R6	Fallas en el servidor	TÉCNICO	Suspensión temporal del sistema.
R7	Mal diseño de la Base de datos	TÉCNICO	Retraso en el desarrollo del sistema
R8	Incumplimientos a las fechas de entregas	TÉCNICO	Retraso en la entrega

Realizado por: Wilmer B. 2018.

Análisis de Riesgos

Tabla 23-2: Priorización de riesgos.

Id	Descripción	Probabilidad			Impacto		Exposición	
		Porcentaje	Probabilidad	Valor	Impacto	Valor	Exposición	Valor
R1	No se analizó correctamente los requerimientos	70%	Alta	3	Alto	3	Alta	9
R2	Desconocimiento de las herramientas que se van a utilizar	70%	Alta	3	Alto	3	Alta	6
R3	Cambios de requerimientos por parte del cliente	70%	Alta	3	Alto	3	Alta	6
R7	Mal diseño de la Base de datos	60%	Alta	3	Alto	3	Alta	5
R4	Daños en el equipo hardware de desarrollo	40%	Media	3	Media	3	Media	3
R5	Falta de comunicación con el cliente	40%	Media	2	Alta	3	Media	3
R6	Fallas en el servidor	40%	Media	2	Baja	1	Baja	2
R8	Incumplimientos a las fechas de entregas	10%	Baja	1	Alta	3	Baja	1

Realizado por: Wilmer B. 2018.

Hojas de gestión de riesgos

Tabla 24-2: Tabla de gestión de riesgo 1.

HOJA DE GESTIÓN DEL RIESGO			
ID. DEL RIESGO: R1		**FECHA:** 04/10/17	
Probabilidad: Alta Valor: 3	**Impacto:** Alto Valor: 3	**Exposición:** Alta Valor: 9	**Prioridad:** 1
DESCRIPCIÓN: No se analizó correctamente los requerimientos			
REFINAMIENTO **Causas** - Falta de comunicación con el cliente - No tiene una visión clara por parte de los desarrolladores. **Consecuencias** - Retraso con el desarrollo del proyecto debido a los cambios de la planificación - El Sistema no cumplirá con los requisitos o funcionalidades del cliente. - Desacuerdos con el usuario.			
REDUCCIÓN - Realizar una reunión con el cliente para poder obtener los requisitos que son necesarios para implementar en el sistema.			
SUPERVISION - Hacer saber al usuario en qué estado se encuentra el desarrollo del sistema. - Mantenerse al tanto de la situación económica y funcional de la empresa.			
GESTIÓN - Tratar de llegar a un acuerdo con el usuario o redefinir tiempos o costes de ser necesario.			

Realizado por: Wilmer B. 2018.

Tabla 25-2: Tabla de gestión de riesgo 2.

HOJA DE GESTIÓN DEL RIESGO			
ID. DEL RIESGO: R2		**FECHA:** 05/10/17	
Probabilidad: Alta Valor: 3	**Impacto:** Alto Valor: 3	**Exposición:** Alta Valor: 6	**Prioridad:** 2
DESCRIPCIÓN: Desconocimiento de las herramientas que se van a utilizar			
REFINAMIENTO **Causas** - Se desconoce las herramientas que se va a utilizar - No se cuente buena documentación para aprender de las herramientas			

Consecuencias - Retraso en la ejecución del proyecto para aprender de la herramienta que se seleccionó para el desarrollo - No se cumple con los tiempos estimados para las entregas de las versiones.
REDUCCIÓN - Realizar capacitaciones en el caso que sea necesario - Tener una buena comunicación con el grupo de trabajo para asegurar el uso adecuado con las herramientas
SUPERVISION - Obtener información necesaria para aprender de las herramientas de desarrollo - Revisiones constantes por parte del jefe del equipo asegurándose de llevar un buen trabajo.
GESTIÓN - Involucrar a personas que conozcan sobre las herramientas

Realizado por: Wilmer B. 2018.

Tabla 26-2: Tabla de gestión de riesgo 3.

HOJA DE GESTIÓN DEL RIESGO			
ID. DEL RIESGO: R3		**FECHA**:05/10/17	
Probabilidad: Alta **Valor:** 3	**Impacto:** Alto **Valor:** 3	**Exposición:** Alta **Valor:** 6	**Prioridad:** 2
DESCRIPCIÓN: Cambios de requerimientos por parte del cliente			
REFINAMIENTO **Causas** - No tuvo una idea clara con los objetivos del sistema - Inconformidad por parte del grupo de desarrollo a los cambios de requerimientos ya teniendo avanzado el sistema. - Malos entendidos **Consecuencias** - Retraso en el desarrollo del sistema para realizar las entregas de versiones. - El sistema no cumple con los requisitos del usuario			
REDUCCIÓN - Obtener todos los requerimientos necesarios del sistema.			
SUPERVISION - Tener un documento escrito para establecer los requerimientos por el usuario y director del proyecto. - Realizar las entregas de los avances del sistema.			

GESTIÓN - Tener un documento que se establezcan los requerimientos del sistema desde el inicio

Realizado por: Wilmer B. 2018.

Tabla 27-2: Tabla de gestión de riesgo 7.

HOJA DE GESTIÓN DEL RIESGO			
ID. DEL RIESGO: R7		**FECHA**: 05/10/17	
Probabilidad: Alta **Valor:** 3	**Impacto:** Alto **Valor:** 3	**Exposición:** Alta **Valor:** 5	**Prioridad:** 3
DESCRIPCIÓN: Mal diseño de la Base de datos			
REFINAMIENTO **Causas** - No se analizó bien los requerimientos - El cliente no se especificó bien con los requisitos **Consecuencias** - Retraso en el desarrollo del sistema. - Retraso en las entregas de las versiones. - Obtener una mala planificación			
REDUCCIÓN - Tener una buena comunicación con el cliente para obtener todos los requerimientos - Analizar los requerimientos			
SUPERVISION - Realizar la normalización de la base de datos			
GESTIÓN - Hacer revisar a personas que tengan experiencia en diseño de Base de datos.			

Realizado por: Wilmer B. 2018.

Tabla 28-2: Tabla de gestión de riesgo 4.

HOJA DE GESTIÓN DEL RIESGO			
ID. DEL RIESGO: R4		**FECHA**: 06/10/17	
Probabilidad: Media **Valor:** 2	**Impacto:** Media **Valor:** 3	**Exposición:** Media **Valor:** 3	**Prioridad:** 5
DESCRIPCIÓN: Daños en el equipo hardware de desarrollo			
REFINAMIENTO **Causas** - Se daña los equipos que se usa para el desarrollo **Consecuencias** - Suspensión parcial o total del proyecto. - Pérdida parcial o total de la información. - Perdida financiera.			
REDUCCIÓN - Realizar un respaldo de toda la información como el proyecto - Instalar antivirus para los equipos de desarrollo.			
SUPERVISION - Verificar contantemente que los equipos no muestren fallas.			
GESTIÓN - Comprar repuestos originales para no tener inconvenientes después.			

Realizado por: Wilmer B. 2018.

Tabla 29-2: Tabla de gestión de riesgo 5.

HOJA DE GESTIÓN DEL RIESGO			
ID. DEL RIESGO: R5		**FECHA**: 06/10/17	
Probabilidad: Media **Valor: 2**	**Impacto:** Alta **Valor:** 3	**Exposición:** Media **Valor:** 3	**Prioridad:** 6
DESCRIPCIÓN: Falta de comunicación con el usuario			
REFINAMIENTO **Causas** - No se tiene una buena comunicación con el cliente - No hay confianza y poco contacto con el cliente **Consecuencias** - No tienen una visión clara sobre el objetivo del proyecto. - Requisitos mal planteados			

REDUCCIÓN - Reuniones constantes con el cliente. - Crear un ambiente amigable y agradable entre el usuario y los desarrolladores.
SUPERVISION - Mantener información al cliente sobre el proceso del proyecto
GESTIÓN - Asegurarse que el cliente se involucre también en el proyecto.

Realizado por: Wilmer B. 2018.

Tabla 30-2: Tabla de gestión de riesgo 8.

HOJA DE GESTIÓN DEL RIESGO			
ID. DEL RIESGO: R8		**FECHA**: 06/10/17	
Probabilidad: Baja **Valor:** 1	**Impacto:** Alta **Valor:** 3	**Exposición:** Baja **Valor:** 1	**Prioridad:** 9
DESCRIPCIÓN: Incumplimientos a las fechas de entregas			
REFINAMIENTO **<u>Causas</u>** - Falta de trabajo en equipo - Mala estimación de las actividades en la planificación - Desconocimiento de las herramientas del sistema **<u>Consecuencias</u>** - Retraso en el desarrollo del sistema - Incumplimientos a las fechas de entregas			
REDUCCIÓN - Asegurar que las personas de desarrollo tengan conocimientos con las herramientas de desarrollo que se van a utilizar. - Que todos los equipos de trabajo se comprometan con el desarrollo del sistema.			
SUPERVISION - Supervisión con los avances del proyecto.			
GESTIÓN - Capacitar al personal para puedan utilizar las herramientas de desarrollo. - Asegurar una comunicación con todos los integrantes del equipo.			

Realizado por: Wilmer B. 2018.

Tabla 31-2: Tabla de gestión de riesgo 6.

HOJA DE GESTIÓN DEL RIESGO			
ID. DEL RIESGO: R6		**FECHA**: 06/10/17	
Probabilidad: Media **Valor: 2**	**Impacto:** Baja **Valor:** 1	**Exposición:** Baja **Valor:** 2	**Prioridad:** 8
DESCRIPCIÓN: Fallas en el servidor			
REFINAMIENTO **Causas** - No se tenga acceso con el servidor cuando se requiera - Tener bien configurado el sistema con el servidor **Consecuencias** - Incumplimiento con las fechas establecidas - No se entregue a tiempo los avances			
REDUCCIÓN - Asegurarse que el servidor este bien configurado con el sistema			
SUPERVISION - Dar mantenimiento periódico al servidor			
GESTIÓN - Asignar a una persona quien de mantenimiento al sistema.			

Realizado por: Wilmer B. 2018.

Tabla 32-2: Tabla prueba de aceptación 02.

PRUEBA DE ACEPTACIÓN	
Id: PA_02	**Nombre:** Verificar que las posibles amenazas o riesgos en el proyecto tengan un plan de contingencia.
Tarea de ingeniería: TI_01 Identificación y probabilidad de riesgos	
Descripción: Para llevar a cabo un proyecto exitoso fue ha desarrollado la gestión de los riesgos probables a ocurrir.	
Responsable: Wilmer Barrera	**Fecha:** 06/10/17
Condición de ejecución: - Tener realizada una planificación del proyecto - Tener definidos los usuarios y roles involucrados en el proyecto	
Pasos de ejecución: - Verificar todos los posibles riesgos de las personas involucradas en el proyecto	

- Crear plan de contingencia para cada uno de ellos.	
Resultado esperado: Cada uno de los posibles riesgos has sido identificados y tratados	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

2.10.1.3 HT_03 Diseño de base de datos

Tabla 33-2: Tabla historia técnica 03.

HISTORIA TÉCNICA		
Id: HT_03	**Nombre:** Diseño de base de datos	
Descripción: Como programador necesito diseñar la base de datos partiendo de los requerimientos. Así como realizar distintos tipos de modelos para llevar una mejor organización de esta.		
Usuario: Wilmer Barrera	**Sprint:** 1	
Fecha inicio: 09/10/17	**Fecha fin:** 13/10/17	**Esfuerzo:** 25

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Diseño de modelo entidad relación
TI_02	Diseño de diagrama entidad relación
TI_03	Diseño de diagrama de clases.
TI_04	Realización de diccionario de datos

Realizado por: Wilmer B. 2018.

Tabla 34-2: Tabla tarea de ingeniería 01 de historia técnica 03.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** diseño	**Sprint:** 1
Nombre de historia técnica: HT_03 Diseño de base de datos		
Nombre tarea: Diseño de modelo entidad relación.		
Fecha inicio: 09/10/17		**Fecha fin:**10/10/17
Descripción: Como desarrollador necesito realizar el diseño principal de la base de datos el cual se basa en los requerimientos del sistema, representándolo a través de un modelo entidad relación		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_03	Verificar que el modelo entidad relación cumpla con los requerimientos y necesidades que planteó el cliente

Realizado por: Wilmer B. 2018.

El presente diseño contiene un conjunto de 6 entidades relacionadas entre sí, estas entidades son las encargadas de gestionar la información que puede variar comúnmente dentro de la información concerniente a horarios, cooperativas, rutas, terminales, etc.

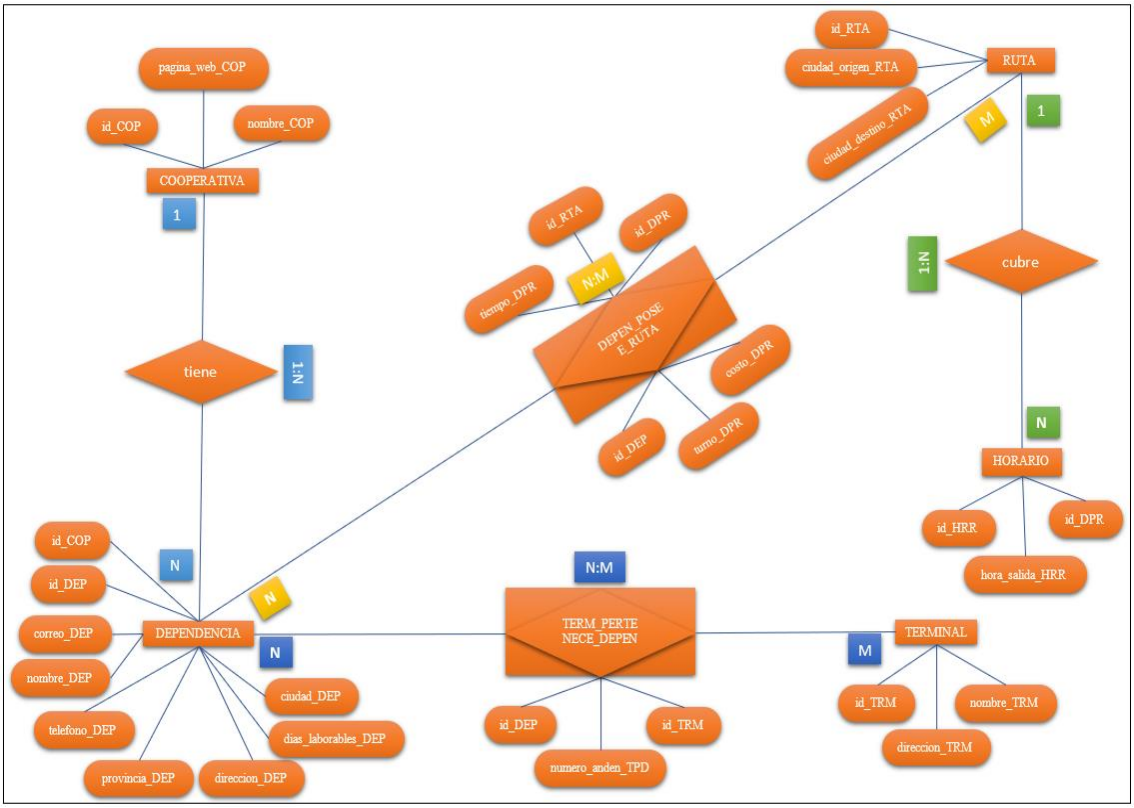


Gráfico 2-2: Modelo entidad relación – 6 entidades en las que la información varía constantemente.

Realizado por: Wilmer B. 2018.

También dentro del modelo entidad relación existe un conjunto de 2 entidades que se encuentran relacionadas entre sí, estas funcionarán como tablas precargadas en la base de datos tanto en el servicio de hosting como en la base de datos local. Estas entidades se usarán específicamente para el filtrado de datos geolocalizados dentro del área delimitada de Ecuador.

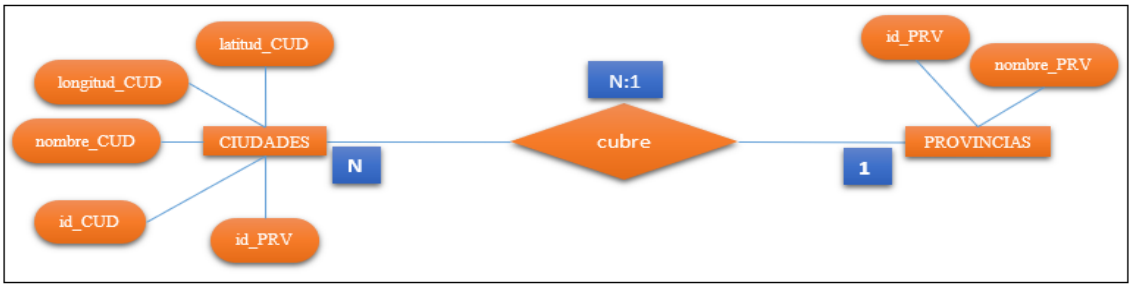


Gráfico 3-2: Modelo entidad relación - 2 entidades en las que la información se persiste.

Realizado por: Wilmer B. 2018.

Por último, se ha creado una entidad sin relación alguna con el resto de las entidades. Esto debido a que se requirió de la funcionalidad de implementación de favoritos dentro de recorridos de rutas. Como el servicio que se preste con la app móvil para los usuarios finales no se brindará a través de cuentas de usuario, entonces se ha optado por agregar una entidad sin relación la cual manejará dichos registros.

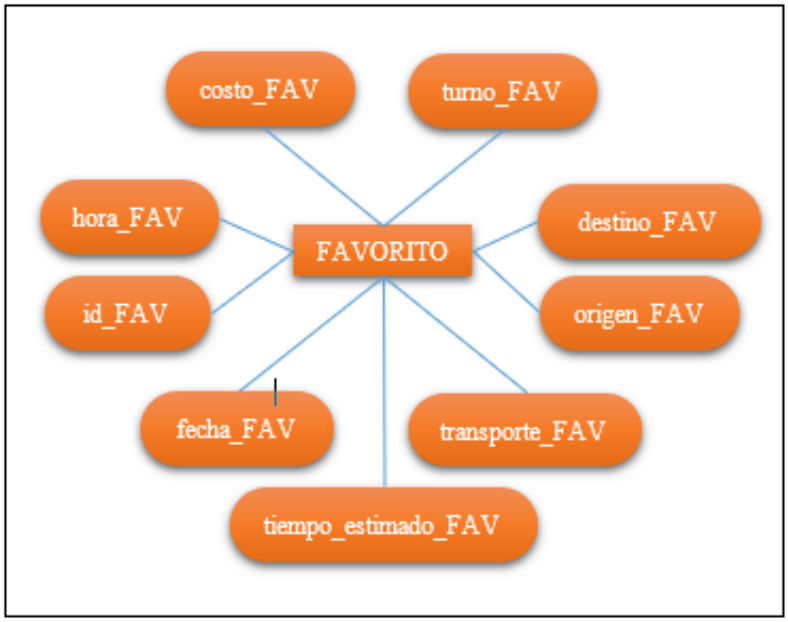


Gráfico 4-2: Modelo entidad relación - 1 entidad no relacionada.

Realizado por: Wilmer B. 2018.

Tabla 35-2: Tabla prueba de aceptación 03.

PRUEBA DE ACEPTACIÓN	
Id: PA_03	**Nombre:** Verificar que el modelo entidad relación cumpla con los requerimientos y necesidades que planteó el cliente
Tarea de ingeniería: TI_01 Diseño de modelo entidad relación.	
Descripción: La estructura de la base de datos debe ser acorde con los requerimientos establecidos.	
Responsable: Wilmer Barrera	**Fecha:** 10/10/17
Condición de ejecución: - Tener la lista de requerimientos. - Separar requerimientos funcionales. - Separar requerimientos no funcionales.	
Pasos de ejecución: - Definir la lista de entidades. - Definir campos establecidos para dichas entidades.	

- Formas relaciones entre entidades. - Definir el tipo de relación entre entidades.
Resultado esperado: El modelo debe contener tablas los campos que satisfagan los requerimientos del cliente.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla 36-2: Tabla tarea de ingeniería 02 de historia técnica 03.

TAREA DE INGENIERÍA		
Id: TI_02	**Tipo de tarea:** diseño	**Sprint:** 1
Nombre de historia técnica: HT_03 Diseño de base de datos		
Nombre tarea: Diseño de diagrama entidad relación.		
Fecha inicio: 11/10/17		**Fecha fin:**11/10/17
Descripción: Partiendo de modelo entidad relación se procederá a representar el mismo a través de un diagrama entidad relación, el cual se acerca mucho más a cómo será definida la base de datos en el sistema.		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_04	Verificar que el diagrama entidad relación siga el modelo entidad relación en el cual se basará este proyecto	

Realizado por: Wilmer B. 2018.

A continuación, siguiendo con el proceso de modelado de datos, partiendo del diagrama entidad relación se procedió a realizar el modelado del diagrama entidad relación.

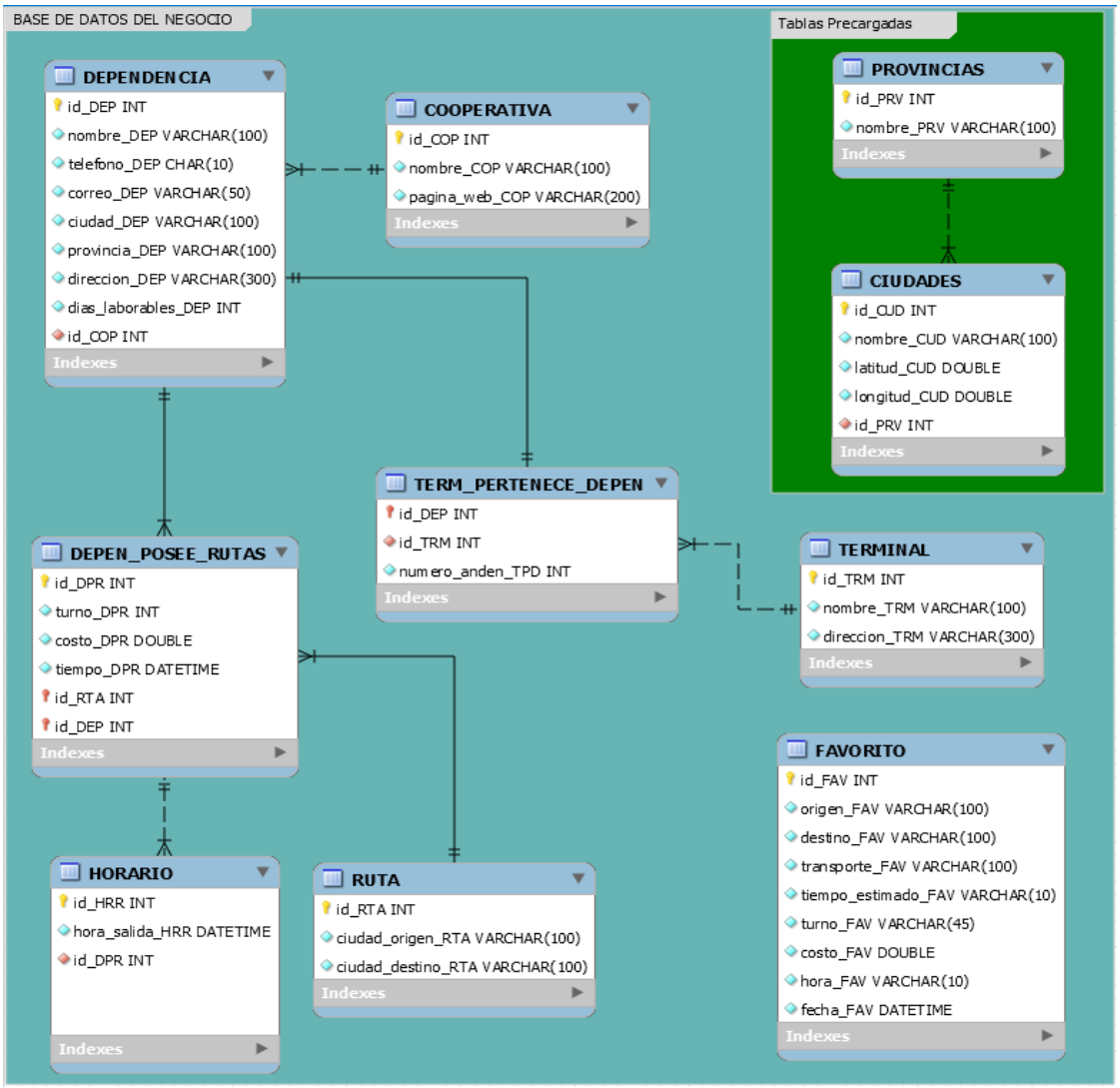


Gráfico 5-2: Modelo entidad relación.

Realizado por: Wilmer B. 2018.

Tabla 37-2: Tabla prueba de aceptación 04.

PRUEBA DE ACEPTACIÓN	
Id: PA_04	**Nombre:** Verificar que el diagrama entidad relación siga el modelo entidad relación en el cual se basará este proyecto
Tarea de ingeniería: TI_02 Diseño de diagrama entidad relación	
Descripción: Siguiendo el diseño de base de datos se procederás a realizar el diagrama entidad relación.	
Responsable: Wilmer Barrera	**Fecha:** 11/10/17
Condición de ejecución: - Tener realizado el modelo entidad relación.	
Pasos de ejecución:	

- Verificar que cada entidad esté relacionada - Verificar que cada campo se ajuste a su requerimiento.	
Resultado esperado: Diagrama entidad relación satisface las necesidades requeridas por el cliente en el diseño de la base de datos.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla 38-2: Tabla tarea de ingeniería 03 de historia técnica 03.

TAREA DE INGENIERÍA		
Id: TI_03	**Tipo de tarea:** diseño	**Sprint:** 1
Nombre de historia técnica: HT_03 Diseño de base de datos		
Nombre tarea: Diseño de diagrama de clases.		
Fecha inicio: 12/10/17		**Fecha fin:**12/10/17
Descripción: Con el objetivo de llevar la documentación organizada y de una manera entendible para otras personas dedicadas al desarrollo se realizará el diagrama de clases en el cual se define claramente la estructura de la base de datos.		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_05	Verificar que en el diagrama de clases se especifique los campos, tipos de datos y relaciones entre tablas claramente.	

Realizado por: Wilmer B. 2018.

Uno de los pasos en el modelado de datos es el diseño del diagrama de clases el cual se lo ha realizado con la herramienta StarUML, siguiendo el modelado de datos del diagrama entidad relación.

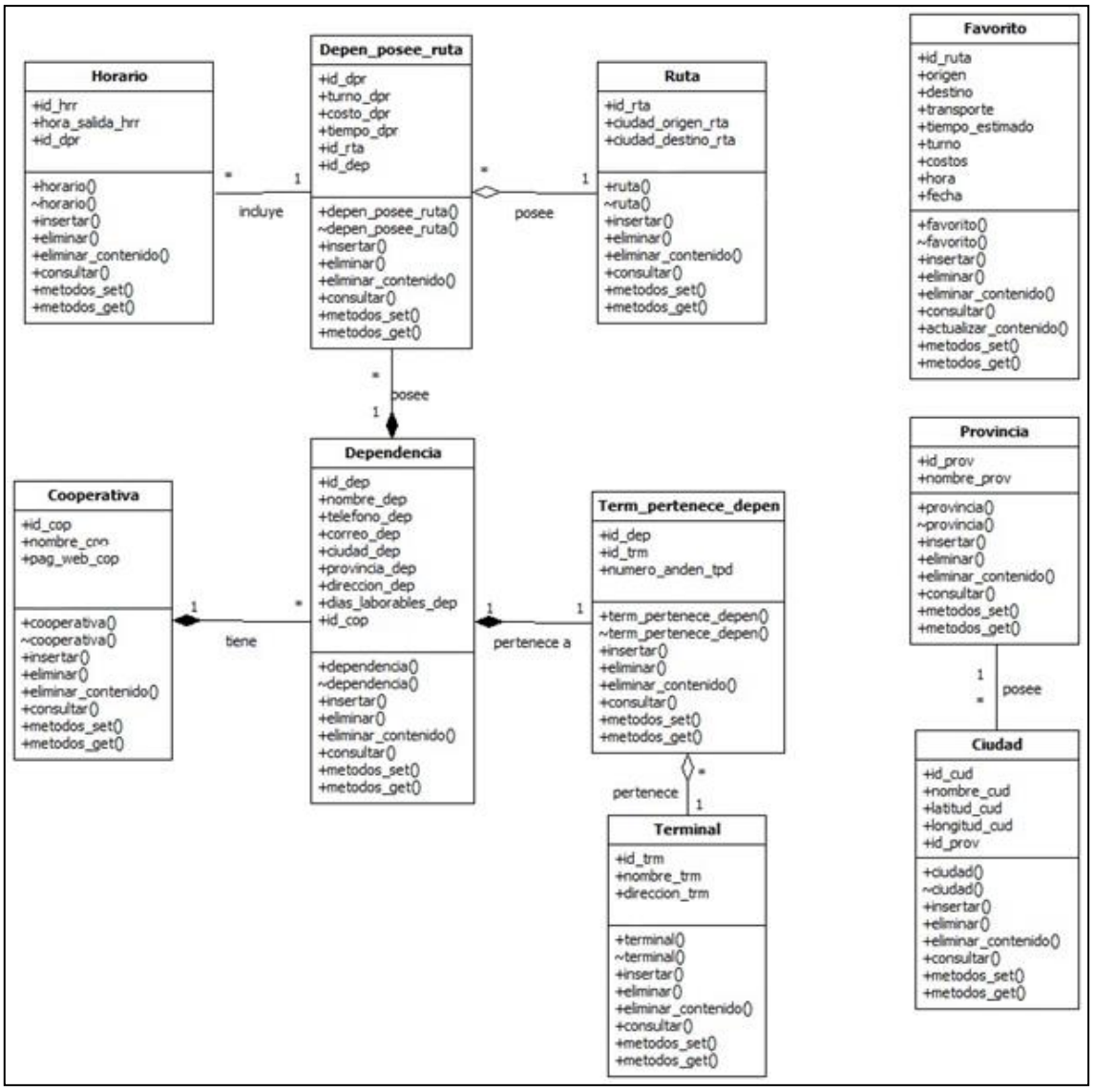


Gráfico 6-2: Diagrama de clases.

Realizado por: Wilmer B. 2018.

El modelo entidad relación la cual funcionará de manera local dentro de la app móvil, dicha base de datos usa el gestor de base de datos SQLite.

Tabla 39-2: Tabla prueba de aceptación 05.

PRUEBA DE ACEPTACIÓN	
Id: PA_05	**Nombre:** Verificar que en el diagrama de clases se especifique los campos, tipos de datos y relaciones entre tablas claramente.
Tarea de ingeniería: TI_03 Diseño de diagrama de clases	
Descripción: El diagrama de clases mostrar a la estructura de la base de datos, cómo está compuesta y definida.	
Responsable: Wilmer Barrera	**Fecha:** 12/10/17

Condición de ejecución: - Tener listo el modelo entidad relación. - Tener listo el diagrama entidad relación.	
Pasos de ejecución: - Verificar qué cada entidad sea representa en una tabla - Verificar que los tipos campos en definidos de manera correcta - Verificar que se describen los métodos necesarios en cada tabla - Verificar las relaciones existentes entre cada tabla.	
Resultado esperado: El diagrama de clases ha sido estructurado correctamente de manera clara y precisa.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla 40-2: Tabla tarea de ingeniería 04 de historia técnica 03.

TAREA DE INGENIERÍA		
Id: TI_04	**Tipo de tarea:** diseño	**Sprint:** 1
Nombre de historia técnica: HT_03 Diseño de base de datos		
Nombre tarea: Realización de diccionario de datos.		
Fecha inicio: 13/10/17		**Fecha fin:**13/10/17
Descripción: Para llevar una documentación óptima sobre la estructura de la base de datos y la definición de campos de esta, así como la representación de valores por defecto en dichos campos se realiza el diccionario de datos.		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_06	Verificar que se haya especificado cada uno de los tipos de datos utilizados en los campos de tablas en las bases de datos y también la representación de los valores por defecto que contienen los campos.	

Realizado por: Wilmer B. 2018.

En la siguiente tabla se detalla a continuación la primera parte del diccionario de datos en esta se detalla campos generales y específicos junto con el tipo de dato asignado y un tamaño o longitud.

Tabla 41-2: Diccionario de datos - Campos y tipos.

CAMPO	TIPO DE DATO	LONGITUD
correo	varchar	50
cedula	char	10
nombre persona	varchar	60
apellido	varchar	60
password	varchar	50
telefono	char/varchar	10 u 11
ciudad	varchar	50
provincia	varchar	50
comentarios	varchar	1000
nombre otros	varchar	100
página web	varchar	200
hora	time	
fecha	date	
direccion	varchar	300

Realizado por: Wilmer B. 2018.

Por otra parte, en la siguiente tabla se detallas valores por defecto o que serán asignados de acuerdo a un tipo o clase dentro de un campo, estos valores representarán información entendible para el usuario, los mismos que serán mostrados al usuario final para un mejor entendimiento.

Tabla 42-2: Diccionario de datos - Representación de campos.

CAMPO	VALOR	REPRESENTACION
logo cooperativa	0	con logo
	1	sin logo
comentario coop	null	sin registro
página web	null	sin registro
telefono_usuaio	null	sin registro
numero anden	0	sin registro
turno	1	normal
	2	ejecutivo
	3	extras
días laborables, todos los dígitos seguidos q forman un único número entero	1	lunes
	2	martes
	3	miércoles
	4	jueves
	5	viernes
	6	sábado
	7	domingo
tipo boleto	1	normal

	2	3ra edad
	3	especial
	4	niños
estado boleto	0	vendido
	1	libre
estado contrato	0	caducado
	1	vigente
estados generales	0	eliminación lógica
	1	tabla activa

Realizado por: Wilmer B. 2018.

Tabla 43-2: Tabla prueba de aceptación 06.

PRUEBA DE ACEPTACIÓN	
Id: PA_06	**Nombre:** Verificar que se haya especificado cada uno de los tipos de datos utilizados en los campos de tablas en las bases de datos y también la representación de los valores por defecto que contienen los campos.
Tarea de ingeniería: TI_04 Realización de diccionario de datos	
Descripción: El diccionario de datos mostrará los detalles de definición de los campos de tablas, como también el tipo de campo y los valores representativos que se asignan a un campo específico.	
Responsable: Wilmer Barrera	**Fecha:** 13/10/17
Condición de ejecución: - Haber diseñado el modelo entidad relación. - Haber diseñado el diagrama entidad relación. - Haber diseñado el diagrama de clases.	
Pasos de ejecución: - Verificar que los campos tengan la longitud de datos correcto - Verificar que cada campo corresponde a un tipo para el que está definido dicho dato - Verificar descripción en cada campo representativo.	
Resultado esperado: El diccionario de datos encuentra detallado correctamente.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

2.10.2 Sprint 2

Tabla 44-2: Tabla de actividades del sprint 02.

ACTIVIDADES SPRINT 02			
Id	**Historia técnica/de usuario**	**Fecha**	**Estimación**
HU_01	Codificación de scripts SQL para transacciones en base de datos	16/10/17 - 25/10/17	40
HT_06	Implementación de pruebas automatizadas Sprint 2	26/10/17 - 27/10/17	10
		16/10/17 - 27/10/17	50

Realizado por: Wilmer B. 2018.

Para el presente sprint se ha realizado un total de 2 historias, una técnica y una de usuario, 4 tareas de ingeniería y 11 pruebas manuales, con un total de 50 puntos. Llevándose a cabo desde el 16/10/17 hasta 27/10/17 siguiendo la planificación antes definida. La documentación completa que sustenta el desarrollo de las actividades se encuentra en el **Anexo B**.

2.10.3 Sprint 3

Tabla 45-2: Tabla de actividades del sprint 03.

ACTIVIDADES SPRINT 03			
Id	**Historia técnica/de usuario**	**Fecha**	**Estimación**
HT_04	Diseño de arquitectura de la aplicación móvil	30/10/17 - 30/10/17	5
HU_02	Implementación de interfaz para consulta de horarios por cooperativa de transporte	31/10/17 - 01/11/17	10
HU_03	Implementación de servicios web Geocoding API de Google Maps a través de petición GET	02/11/17 - 08/11/17	25
HT_07	Implementación de pruebas automatizadas Sprint 3	09/11/17 - 10/11/17	10
		30/10/17 - 10/11/17	50

Realizado por: Wilmer B. 2018.

Para el presente sprint se ha realizado un total de 4 historias, 2 historias técnicas y dos de usuario, 6 tareas de ingeniería y 12 pruebas manuales, con un total de 50 puntos. Llevándose a cabo desde el 30/10/17 hasta 10/11/17 siguiendo la planificación antes definida. La documentación completa que sustenta el desarrollo de las actividades se encuentra en el **Anexo B**.

2.10.3.1 HT_04 Diseño de arquitectura de la aplicación móvil

Tabla 46-2: Tabla historia técnica 04.

HISTORIA TÉCNICA		
Id: HT_04	**Nombre:** Diseño de arquitectura de la aplicación móvil	
Descripción: Como programador necesito analizar y diseñar la arquitectura más acorde con el proyecto, así como también como está compuesto y desplegado el mismo.		
Usuario: Wilmer Barrera	**Sprint:** 3	
Fecha inicio: 30/10/17	**Fecha fin:** 30/10/17	**Esfuerzo:** 5

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Implementación de diseño de arquitectura del sistema
TI_02	Diseño de diagrama de componentes
TI_03	Diseño de diagrama de despliegue

Realizado por: Wilmer B. 2018.

Tabla 47-2: Tabla tarea de ingeniería 01 de historia técnica 04.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** diseño	**Sprint:** 3
Nombre de historia técnica: HT_04 Diseño de arquitectura de la aplicación móvil		
Nombre tarea: Implementación de diseño de arquitectura del sistema.		
Fecha inicio: 30/10/17		**Fecha fin:**30/10/17
Descripción: Como desarrollador necesito realizar la representación gráfica de la arquitectura seleccionada en la que se realizará este proyecto		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_18	Verificar que la arquitectura seleccionada sea la que mejor se adapte al requerimiento del proyecto.

Realizado por: Wilmer B. 2018.

La arquitectura es parte esencial dentro de la primera fase del desarrollo de un proyecto software, para este proyecto se ha definido una arquitectura cliente-servidor. El cliente vendría a ser la aplicación móvil que se encuentra instalada en los teléfonos de los usuarios finales, mientras que tenemos dos servidores: Servidor de Google Maps, este se utiliza para realizar el proceso de localización de un dispositivo dentro del área delimitada de Ecuador, dándonos como resultados

un conjunto de respuestas geo codificadas las mismas que serán procesadas con el objetivo de presentar resultados de fácil entendimiento para el usuario.

Otro de los servidores se encuentra ubicado en un servicio de hosting el cual se encargada de almacenar la información en la base de datos como también en gestionar las peticiones realizadas desde los dispositivos móviles a través de los servicios web almacenados en este.

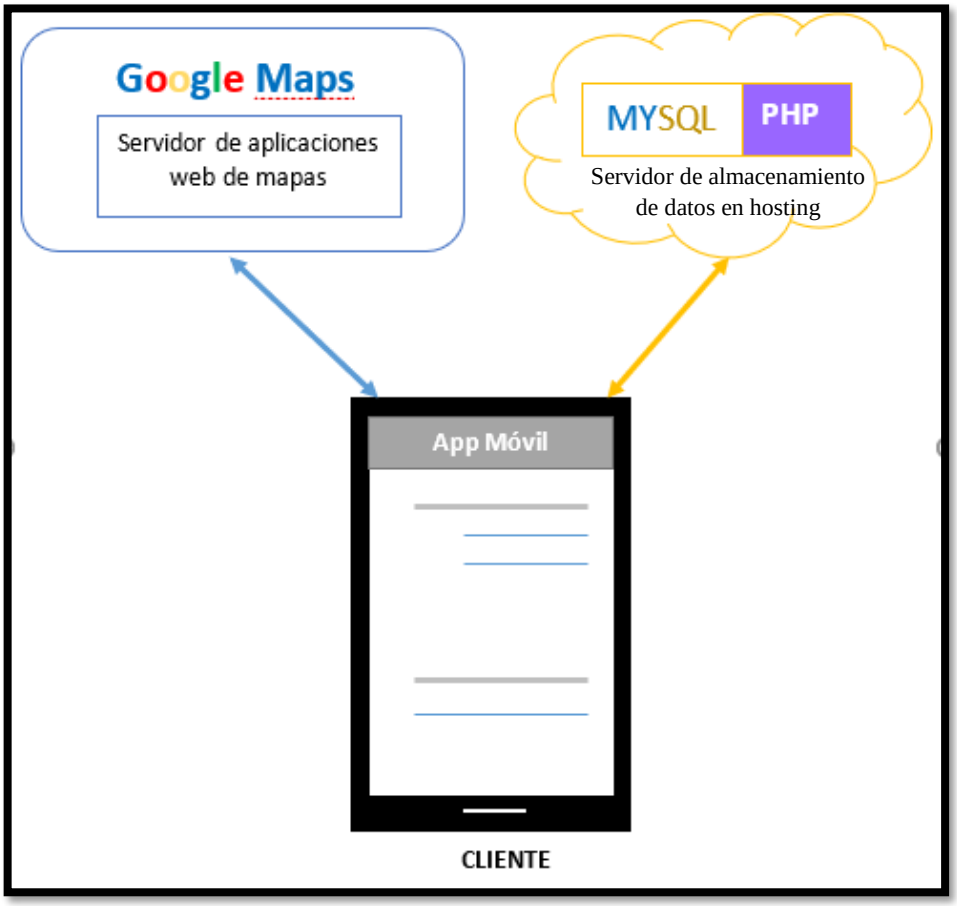


Figura 4-2: Arquitectura del sistema.

Realizado por: Wilmer B. 2018.

Tabla 48-2: Tabla prueba de aceptación 18.

PRUEBA DE ACEPTACIÓN	
Id: PA_18	**Nombre:** Verificar que la arquitectura seleccionada sea la que mejor se adapte al requerimiento del proyecto.
Tarea de ingeniería: TI_01 Implementación de diseño de arquitectura del sistema	
Descripción: La arquitectura del sistema será verificada de manera que cumpla con los requerimientos establecidos con el cliente.	
Responsable: Wilmer Barrera	**Fecha:** 30/10/17

Condición de ejecución: - Tener definido los requerimientos del sistema
Pasos de ejecución: - Identificar cada uno de los nodos por los que navegará la información - Verificar que la arquitectura sea la óptima de acuerdo a los recursos que tiene el cliente
Resultado esperado: De acuerdo a los recursos que dispone el cliente y los requerimientos establecidos con el mismo, se he establecido satisfactoriamente la arquitectura en la que se implementará la app móvil.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla 49-2: Tabla tarea de ingeniería 02 de historia técnica 04.

TAREA DE INGENIERÍA		
Id: TI_02	**Tipo de tarea:** diseño	**Sprint:** 3
Nombre de historia técnica: HT_04 Diseño de arquitectura de la aplicación móvil		
Nombre tarea: Diseño de diagrama de componentes.		
Fecha inicio: 30/10/17		**Fecha fin:**30/10/17
Descripción: Como desarrollador necesito dar a conocer los componentes a usar en la aplicación móvil, esto representado a través de un diagrama de componentes.		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_19	Verificar que todos los componentes involucrados en este proyecto sean representados de manera clara.	

Realizado por: Wilmer B. 2018.

Uno de los procesos dentro de la fase de diseño es recomendable usar o presentar el diagrama de componentes ya que esta muestra la estructura y los niveles de los componentes que actúan en la aplicación. A continuación, se presenta dicho diagrama.

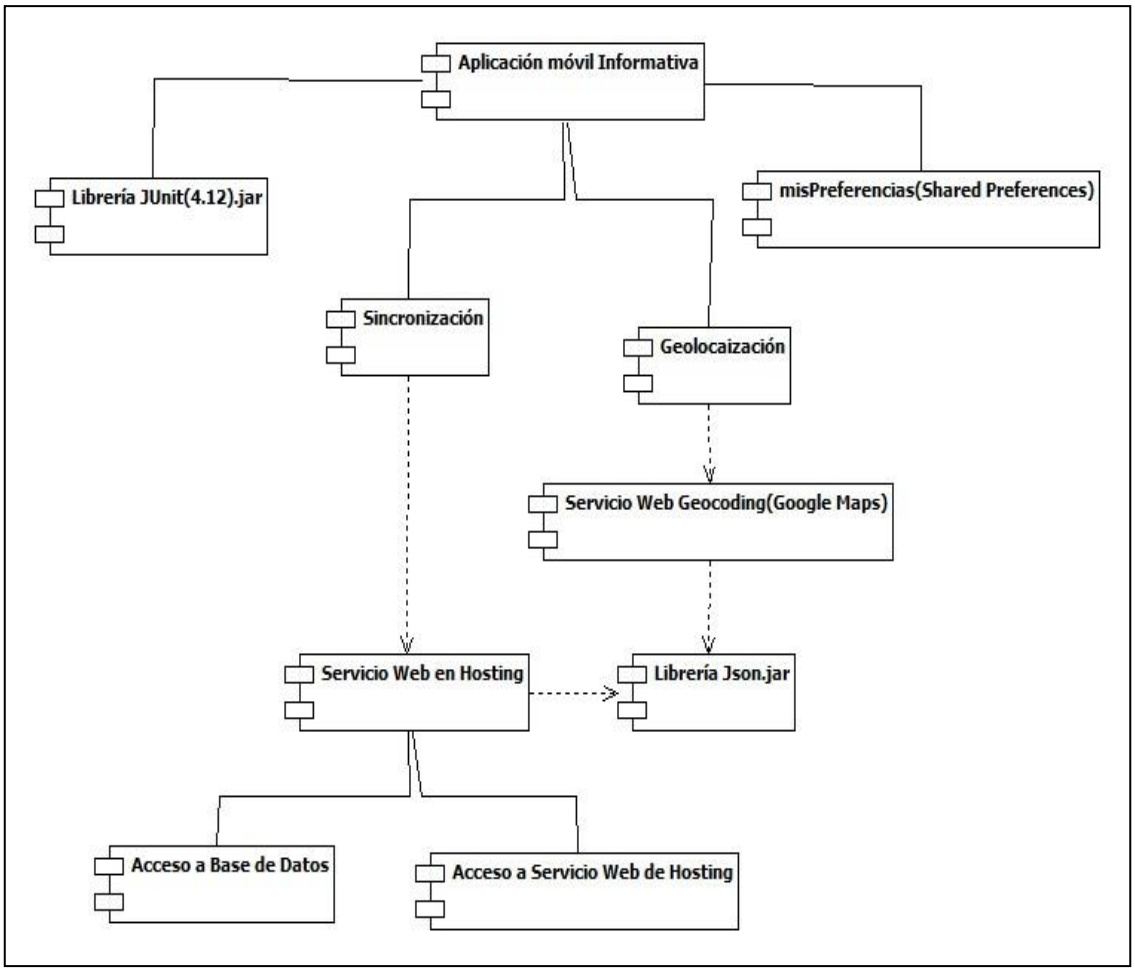


Gráfico 7-2: Diagrama de componentes del sistema.

Realizado por: Wilmer B. 2018.

Tabla 50-2: Tabla prueba de aceptación 19.

PRUEBA DE ACEPTACIÓN	
Id: PA_19	**Nombre:** Verificar que todos los componentes involucrados en este proyecto sean representados de manera clara.
Tarea de ingeniería: TI_02 Diseño de diagrama de componentes	
Descripción: cada uno de los componentes que se usan en la aplicación deben ser descritos en forma de esquema.	
Responsable: Wilmer Barrera	**Fecha:** 30/10/17
Condición de ejecución: - Tener definida la arquitectura del sistema - Tener definida la plataforma en la que se desarrollará la aplicación móvil	
Pasos de ejecución: - Verificar que componentes se usarán en l arquitectura del sistema	

- Verificar que todos los componentes que integran el sistema se encuentren representados en un esquema de componentes.
Resultado esperado: Se debe mostrar una estructura esquemática entendible y clara de cómo interactúan los componentes entre si dentro del sistema.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla 51-2: Tabla tarea de ingeniería 03 de historia técnica 04.

TAREA DE INGENIERÍA		
Id: TI_03	**Tipo de tarea:** diseño	**Sprint:** 3
Nombre de historia técnica: HT_04 Diseño de arquitectura de la aplicación móvil		
Nombre tarea: Diseño de diagrama de despliegue.		
Fecha inicio: 30/10/17		**Fecha fin:**30/10/17
Descripción: Como desarrollador necesito dar a conocer cómo funcionará el sistema una vez que se ejecute en producción, dando así a conocer los distintos nodos por los cuales pasará la información a usarse en la aplicación móvil		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_20	Verificar que se muestre la estructura por los que navegará la aplicación una vez desplegado el sistema.	

Realizado por: Wilmer B. 2018.

Uno de los diagramas importantes es el que se presenta a continuación, conocido como diagrama de despliegue en el cual se puede ver la arquitectura del sistema en tiempo de ejecución, así como el flujo que en que se moverán componentes y/o información en los diferentes nodos del sistema desplegado.

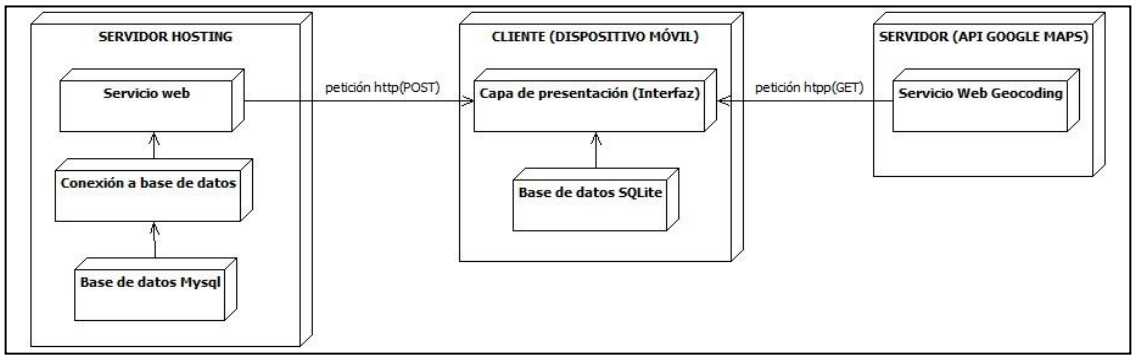


Gráfico 8-2: Diagrama de despliegue del sistema.

Realizado por: Wilmer B. 2018.

Tabla 52-2: Tabla prueba de aceptación 20.

PRUEBA DE ACEPTACIÓN	
Id: PA_20	**Nombre:** Verificar que se muestre la estructura por los que navegará la aplicación una vez desplegado el sistema.
Tarea de ingeniería: TI_03 Diseño de diagrama de despliegue	
Descripción: Se representará la estructura que sigue el sistema en tiempo de ejecución	
Responsable: Wilmer Barrera	**Fecha:** 30/10/17
Condición de ejecución: - Tener establecida la arquitectura del sistema - Tener definido el sistema gestor de base de datos - Tener los clientes del sistema	
Pasos de ejecución: - Verificar que se representen todos los servidores a utilizar - Verificar el esquema donde se representen el alojamiento de los servicios web - Verificar el esquema donde el cliente principal consumirá los servicios web del servidor.	
Resultado esperado: Se mostrará el flujo que sigue el sistema en el que se basa la app móvil	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Codificación

En el desarrollo de la aplicación móvil informativa sobre horarios de rutas del transporte público y/o privado se ha desarrollado en un total de 10 Sprints, los mismos que se han entregado según el plan de entrega propuesto.

La documentación de este proyecto se encuentra en el **Anexo B** de este documento, el mismo que contiene las respectivas historias tanto técnicas como de usuario, y a la vez cada una de estas tiene sus respectivas tareas de ingeniería y pruebas de aceptación.

En el desarrollo de este proyecto se ha llevado a cabo bajo el lenguaje de programación java 8, en el entorno de desarrollo integrado Android Studio, servidor Wamp Sever para almacenamientos de servicios web creaos en php destinados a usarse como pruebas previas a la implantación del sistema, así como también se usaron los gestores de base de datos MySql y Sqlite.

2.10.4 Sprint 4

Tabla 53-2: Tabla de actividades del sprint 04.

ACTIVIDADES SPRINT 04			
Id	**Historia técnica/de usuario**	**Fecha**	**Estimación**
HU_04	Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps	13/11/17 - 15/11/17	15
HU_05	Implementación de interfaz para consulta de horarios en base a ciudad origen y destino	16/11/17 - 16/11/17	5
HU_06	Implementación de funcionalidad para consulta de horarios por transporte	17/11/17 - 22/11/17	20
HT_08	Implementación de pruebas automatizadas Sprint 4	23/11/17 - 24/11/17	10
		13/11/17 - 24/11/17	50

Realizado por: Wilmer B. 2018.

Para el presente sprint se ha realizado un total de 4 historias, 3 de usuario y 1 técnica, 14 tareas de ingeniería y 34 pruebas manuales, con un total de 50 puntos. Llevándose a cabo desde el 13/11/17 hasta 24/11/17 siguiendo la planificación antes definida. La documentación completa que sustenta el desarrollo de las actividades se encuentra en el **Anexo B**.

2.10.5 Sprint 5

Tabla 54-2: Tabla de actividades del sprint 05.

ACTIVIDADES SPRINT 05			
Id	**Historia técnica/de usuario**	**Fecha**	**Estimación**
HU_07	Implementación de interfaz favoritos	27/11/17 - 28/11/17	10
HU_08	Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino	29/11/17 - 05/12/17	25
HU_09	Implementación de funcionalidad para obtener la próxima salida en base a la hora actual	06/12/17 - 07/12/17	10

HT_09	Implementación de pruebas automatizadas Sprint 5	08/12/17 - 08/12/17	5
		27/11/17 - 08/12/17	50

Realizado por: Wilmer B. 2018.

Para el presente sprint se ha realizado un total de 4 historias, 3 de usuario y 1 técnica, 9 tareas de ingeniería y 20 pruebas manuales, con un total de 50 puntos. Llevándose a cabo desde el 27/11/17 hasta 08/12/17 siguiendo la planificación antes definida. La documentación completa que sustenta el desarrollo de las actividades se encuentra en el **Anexo B**.

2.10.6 Sprint 6

Tabla 55-2: Tabla de actividades del sprint 06.

ACTIVIDADES SPRINT 06			
Id	**Historia técnica/de usuario**	**Fecha**	**Estimación**
HU_10	Creación de servicios web en PHP para sincronización con app móvil	11/12/17 - 15/12/17	25
HU_11	Implementación de funcionalidad para mostrar horarios por días	18/12/17 - 21/12/17	20
HT_10	Implementación de pruebas automatizadas Sprint 6	22/12/17 - 22/12/17	5
		11/12/17 - 22/12/17	50

Realizado por: Wilmer B. 2018.

Para el presente sprint se ha realizado un total de 3 historias, 2 de usuario y 1 técnica, 5 tareas de ingeniería y 8 pruebas manuales, con un total de 50 puntos. Llevándose a cabo desde el 11/12/17 hasta 22/12/17 siguiendo la planificación antes definida. La documentación completa que sustenta el desarrollo de las actividades se encuentra en el **Anexo B**.

2.10.7 Sprint 7

Tabla 56-2: Tabla de actividades del sprint 07.

ACTIVIDADES SPRINT 07			
Id	**Historia técnica/de usuario**	**Fecha**	**Estimación**
HU_12	Implementación de funcionalidad para obtener coordenadas geográficas en Android	26/12/17 - 28/12/17	15

HU_13	Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte	29/12/17 - 05/01/18	25
HT_11	Implementación de pruebas automatizadas Sprint 7	08/01/18 - 09/01/18	10
		26/12/17 - 09/01/18	50

Realizado por: Wilmer B. 2018.

Para el presente sprint se ha realizado un total de 3 historias, 2 de usuario y una técnica, 6 tareas de ingeniería y 34 pruebas manuales, con un total de 50 puntos. Llevándose a cabo desde el 26/12/17 hasta 09/01/18 siguiendo la planificación antes definida. La documentación completa que sustenta el desarrollo de las actividades se encuentra en el **Anexo B**.

2.10.8 Sprint 8

Tabla 57-2: Tabla de actividades del sprint 08.

ACTIVIDADES SPRINT 08			
Id	**Historia técnica/de usuario**	**Fecha**	**Estimación**
HU_14	Implementación de funcionalidad favoritos	10/01/18 - 16/01/18	25
HU_15	Implementación de funcionalidad llamada telefónica y cargar datos de agencia de cooperativa	17/01/18 - 19/01/18	15
HT_12	Implementación de pruebas automatizadas Sprint 8	22/01/18 - 23/01/18	10
		10/01/18 - 23/01/18	50

Realizado por: Wilmer B. 2018.

Para el presente sprint se ha realizado un total de 3 historias, 2 de usuario y 1 técnica, 6 tareas de ingeniería y 12 pruebas manuales, con un total de 50 puntos. Llevándose a cabo desde el 10/01/18 hasta 23/10/18 siguiendo la planificación antes definida. La documentación completa que sustenta el desarrollo de las actividades se encuentra en el **Anexo B**.

2.10.9 Sprint 9

Tabla 58-2: Tabla de actividades del sprint 09.

ACTIVIDADES SPRINT 09			
Id	**Historia técnica/de usuario**	**Fecha**	**Estimación**
HU_16	Consumo de servicios web a través de petición POST	24/01/18 - 24/01/18	5
HU_17	Sincronización de app móvil con base de datos alojado en servicio de hosting	25/01/18 - 01/02/18	30
HT_13	Implementación de pruebas automatizadas Sprint 9	02/02/18 - 06/02/18	15
		24/01/18 - 06/02/18	50

Realizado por: Wilmer B. 2018.

Para el presente sprint se ha realizado un total de 3 historias, 2 de usuario y 1 técnica, 10 tareas de ingeniería y 36 pruebas manuales, con un total de 50 puntos. Llevándose a cabo desde el 24/01/18 hasta 06/02/18 siguiendo la planificación antes definida. La documentación completa que sustenta el desarrollo de las actividades se encuentra en el **Anexo B**.

2.10.10 Sprint 10

Tabla 59-2: Tabla de actividades del sprint 10.

ACTIVIDADES SPRINT 10			
Id	**Historia técnica/de usuario**	**Fecha**	**Estimación**
HU_18	Implementación de interfaz de instalación	07/02/18 - 14/02/18	20
HT_05	Desarrollo de manual de usuario	15/02/18 - 22/02/18	30
		07/02/18 - 22/02/18	50

Realizado por: Wilmer B. 2018.

Para el presente sprint se ha realizado 1 historia técnica y 1 de usuario, 3 tareas de ingeniería y 3 pruebas manuales, con un total de 50 puntos. Llevándose a cabo desde el 07/02/18 hasta 22/02/18 siguiendo la planificación antes definida. La documentación completa que sustenta el desarrollo de las actividades se encuentra en el **Anexo B**.

CAPÍTULO III

3 RESULTADOS Y DISCUCIÓN

3.1 Fase de cierre

3.1.1 Avance del proyecto

Con la planificación realizada y el desarrollo de cada uno de los sprints se ha podido interpretar gráficamente este avance y los percances que se han tenido en el transcurso de este.

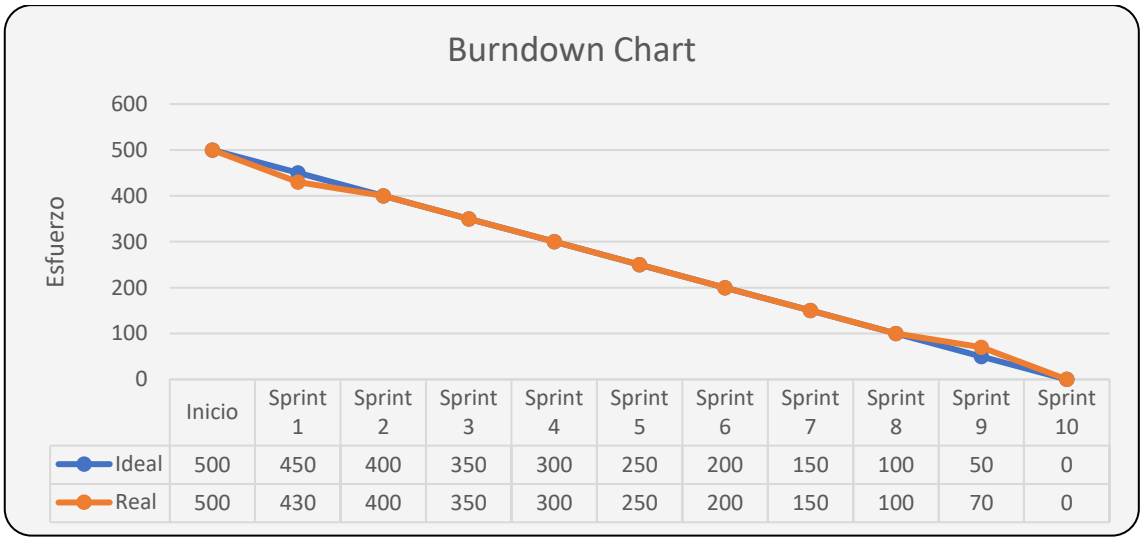


Gráfico 1-3: Burndown chart.

Realizado por: Wilmer B. 2018.

Con un total de 10 sprints planificados en el desarrollo de este proyecto, a través de la gráfica se puede observar una sub y sobrestimaciones en dos sprints en específico, estos son el sprint 1 y el 9. En el sprint 1 hubo una sobrestimación de puntos dentro de los cuales existe una diferencia de 20 puntos entre lo real y lo estimado, esto debido a que no hubo mayor inconveniente en la detección de riesgos que pueden presentarse en este proyecto.

En el sprint 9 hubo una subestimación de puntos con diferencia de 20 puntos entre los reales y estimados, esto debido a que no se tomó en cuenta la realización de filtrado de información concreta de cada tabla que iba a ser almacenada en la base de datos local del dispositivo móvil.

3.1.2 Manual de usuario

El manual de usuario es una guía para la persona encargada en manejar el sistema, este documento se encuentra detallada cada una de las funcionalidades posibles en la aplicación móvil, brindando información concreta y clara a través de imágenes e ilustraciones, detallando así el paso a paso por las distintas interfaces del sistema. Cabe recalcar que este documento se encuentra adjunto al presente trabajo de titulación, en el **Anexo E**. El mismo que está realizado en el editor de textos Microsoft Word, este documento consta de 24 hojas en total, en las que se describe las funcionalidades de la aplicación móvil NIS.

3.2 Resultados de pruebas manuales

En este apartado se presentará cada uno de los resultados de los test manuales realizadas y pasadas a través de pruebas de aceptación.
A continuación, se muestra los tiempos obtenidos en cada una de las pruebas, así como también estas separadas por sprint y por último se muestra un resultado total o general de estos resultados obtenidos.

Síntesis de pruebas de aceptación sprint 2

Tabla 1-3: Resultado de tiempo aplicado en pruebas de aceptación en la HU_01.

Historia de usuario: HU_01 Codificación de scripts SQL para transacciones en base de datos		
Tarea de ingeniería	**Prueba de aceptación**	**Tiempo**
TI_01 Implementación de script para creación de tablas	PA_07 Verificar creación de tabla: COOPERATIVA, DEPENDENCIA, DEPEN_POSEE_RUTAS, HORARIO, RUTA, TERMINAL, TERM_PERTENECE_DEP, PROVINCIAS, CIUDADES, FAVORITO en base de datos local	3s
TI_02 Implementación de script para eliminación de contenido de tablas	PA_10 Verificar que se haya eliminado el contenido de la tabla: COOPERATIVA, DEPENDENCIA, DEPEN_POSEE_RUTAS, HORARIO, RUTA, TERMINAL, TERM_PERTENECE_DEP, PROVINCIAS, CIUDADES, FAVORITO de la base de datos local	3s

TI_03 Implementación de script para eliminación de tablas	PA_12 Verificar que la tabla: COOPERATIVA, DEPENDENCIA, DEPEN_POSEE_RUTAS, HORARIO, RUTA, TERMINAL, TERM_PERTENECE_DEP, PROVINCIAS, CIUDADES, FAVORITO se haya eliminado correctamente de la base de datos local	3s
TI_04 Implementación de funcionalidad para crear base de datos y conexión a esta.	PA_14 Verificar que la base de datos se creó correctamente	4s
Total	4 pruebas funcionales	13s

Realizado por: Wilmer B. 2018.

Las pruebas funcionales correspondientes al sprint 2, tienen un total de 4 pruebas de aceptación aplicadas en una sola historia de usuario, las mismas que tienen un tiempo total de 13 segundos en haber sido procesadas y aprobadas.

Síntesis de pruebas de aceptación sprint 3

Tabla 2-3: Resultado de tiempo aplicado en pruebas de aceptación en la HU_03.

Historia de usuario: HU_03 Implementación de servicios web Geocoding API de Google Maps a través de petición GET		
Tareas de ingeniería	**Pruebas de aceptación**	**Tiempo**
TI_01 Implementación de funcionalidad para realizar peticiones de consumo de web service a través de método get	PA_22 Verificar que haya respuesta del consumo de servicios web de Google Maps	4s
	PA_23 Verificar que consumo de servicio web no fue exitoso	3s
	PA_24 Verificar que consumo de servicio web fue exitoso	5s
	PA_25 Verificar que consumo de servicio web fue exitoso, pero sin resultados	3s
	PA_26 Verificar respuesta de servicio web contiene lista de resultados	5s
	PA_27 Verificar que la conexión al servicio web fue aceptada.	2s
	PA_28 Verificar que la conexión al servicio web fue rechazada.	2s

TI_02 Implementación de petición web en segundo plano.	PA_29 Verificar que ejecución de consumo de servicio web en segundo plano	7s
Total	8 pruebas funcionales	31s

Realizado por: Wilmer B. 2018.

Las pruebas funcionales correspondientes al sprint 3, tienen un total de 8 pruebas de aceptación aplicadas en una sola historia de usuario, estas tienen un tiempo total de 31 segundos en haber sido procesadas y aprobadas.

Síntesis de pruebas de aceptación sprint 4

Tabla 3-3: Resultado de tiempo aplicado en pruebas de aceptación en la HU_04.

Historia de usuario: HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps.		
Tarea de ingeniería	**Prueba de aceptación**	**Tiempo**
TI_01 Implementación de funcionalidad para recolectar resultados de petición a servicio web geocoding	PA_30 Verificar que la lista de resultados no esté vacía	3s
TI_02 Implementación de funcionalidad para filtrar resultado devuelvo de servicio web geocoding	PA_31 Verificar que existe lista de direcciones filtradas.	4s
TI_03 Implementación de funcionalidad para formatear cadenas de direcciones	PA_32 Verificar lista de direcciones fueron formateadas exitosamente	5s
	PA_33 Verificar que cargaron lista de provincias desde la base de datos	3s

TI_04 Implementación de funcionalidad para encontrar coincidencias de provincias del Ecuador con el listado de resultados devuelto por el servicio geocoding	PA_34 Verificar que tabla provincias no se encuentre vacía	3s
	PA_35 Verificar que se encontró coincidencia entre una provincia y el listado de direcciones filtradas	4s
	PA_36 Verificar que no se encontró coincidencia entre una provincia y el listado de direcciones filtradas	3s
TI_05 Implementación de funcionalidad para encontrar coincidencias de ciudades del Ecuador con el listado de resultados devuelto por el servicio geocoding	PA_37 Verificar que se cargue lista de ciudades de una provincia	4s
	PA_38 Verificar que lista de ciudades de una provincia no esté vacía	3s
	PA_39 Verificar cuando se encontró coincidencias de ciudades con la lista de direcciones filtradas	3s
	PA_40 Verificar cuando no se encontró coincidencias de ciudades con la lista de direcciones filtradas	2s
TI_06 Implementación de funcionalidad para obtener la distancia entre dos coordenadas geográficas.	PA_41 Verificación de conversión cantidad normal a radianes	5s
	PA_42 Verificar distancia entre dos coordenadas geográficas	3s
TI_07 Implementar funcionalidad para obtener ciudad más cercana de acuerdo a una dirección específica	PA_43 Verificar que exista índice menor de una lista de ciudades cercana a una dirección específica	4s
	PA_44 Verificar que se cargue toda la lista de ciudades de la base de datos local	3s
	PA_45 Verificar que lista de ciudades extraídas de la base de datos no esté vacía	2s
	PA_46 Verificar si existe una ciudad cercana	73s
	PA_47 Verificar cuando no existe una ciudad cercana	2s
	PA_48 Verificar existencia de una ciudad específica dentro de la tabla CIUDADES	3s
	PA_49 Verificar cuando no existe una ciudad específica dentro de la tabla CIUDADES	3s
Total	20 pruebas funcionales	135s

Realizado por: Wilmer B. 2018.

Tabla 4-3: Resultado de tiempo aplicado en pruebas de aceptación en la HU_06.

Historia de usuario: HU_06 Implementación de funcionalidad para consulta de horarios por transporte		
Tarea de ingeniería	**Prueba de aceptación**	**Tiempo**
TI_01 Crear funcionalidad para extracción de lista de cooperativas de transporte	PA_51 Verificar que se cargó lista de cooperativas de transporte desde la base de datos local	4s
	PA_52 Verificar que lista de cooperativas de transporte esté vacía	3s
TI_02 Crear funcionalidad para extracción de lista de rutas que pertenecen a una cooperativa de transporte	PA_53 Verificar que se cargue lista de rutas de una cooperativa de transporte específica	4s
	PA_54 Verificar que lista de rutas de una cooperativa de transporte específica esté vacía	3s
TI_03 Crear funcionalidad para formatear información de rutas almacenadas en lista de objetos pasando a lista de strings	PA_55 Verificar formateo de campos de objetos a lista de strings	19s
TI_04 Crear funcionalidad para obtener lista de dependencias que pertenecen a una cooperativa de transporte	PA_56 Verificar que se cargue lista de dependencias de una cooperativa de transporte	9s
	PA_57 Verificar existencia de dependencia a la que pertenece una ruta de una cooperativa de transporte	3s
	PA_58 Verificar cuando no existe dependencia a la que pertenece una ruta de una cooperativa de transporte	2s
	PA_59 Convertir lista de objetos de cooperativas de transporte a lista de strings	23s
	PA_60 Verificar cuando no existan dependencias en las que se encuentra registrada una ruta específica	3s
TI_05 Crear funcionalidad para extracción de lista de horarios en base a una ruta de una agencia de cooperativa de transporte específica.	PA_61 Verificar que se cargue lista de horarios de una ruta que pertenece a una agencia de cooperativa de transporte	4s
	PA_62 Verificar cuando la lista de horarios de una ruta que pertenece a una agencia de cooperativa de transporte está vacía	3s
Total	12 pruebas funcionales	80s

Realizado por: Wilmer B. 2018.

Las pruebas funcionales correspondientes al sprint 4, tienen un total de 32 pruebas de aceptación aplicadas y distribuidas en 2 historias de usuario, estas tienen un tiempo total de 215 segundos en haber sido procesadas y aprobadas.

Síntesis de pruebas de aceptación sprint 5

Tabla 5-3: Resultado de tiempo aplicado en pruebas de aceptación en la HU_08.

Historia de usuario: HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino		
Tarea de ingeniería	**Prueba de aceptación**	**Tiempo**
TI_01 Creación de funcionalidad para extraer lista de ciudades origen desde la base de datos	PA_65 Verificar que se cargue lista de ciudades origen desde la base de datos local	3s
	PA_66 Verificar que la lista de ciudades origen esté vacía	3s
TI_02 Creación de funcionalidad para extraer lista de ciudades destino basado en una ciudad origen	PA_67 Verificar que se cargue lista de ciudades destino con base en una ciudad origen desde la base de datos local	4s
	PA_68 Verificar que la lista de ciudades destino esté vacía	4s
TI_03 Crear funcionalidad para formatear lista de objetos ruta a lista de strings	PA_69 Verificar que lista de objetos de rutas sea formateado correctamente a lista de strings	5s
	PA_70 Verificar formateo de campos de rutas empiecen con letra mayúscula	3s
TI_04 Crear funcionalidad para extraer lista de horarios en base a una ruta, sin importar la cooperativa de transporte	PA_71 Verificar que se cargue lista de horarios de una ruta específica	5s
	PA_72 Verificar que lista de horarios de una ruta específica esté vacía	4s
	PA_73 Verificar cuando existe id de una ruta específica	3s
	PA_74 Verificar cuando no existe id de una ruta específica	3s
Total	10 pruebas funcionales	37s

Realizado por: Wilmer B. 2018.

Tabla 6-3: Resultado de tiempo aplicado en pruebas de aceptación en la HU_09.

Historia de usuario: HU_09 Implementación de funcionalidad para obtener la próxima salida en base a la hora actual		
Tarea de ingeniería	**Prueba de aceptación**	**Tiempo**
TI_02 Implementar funcionalidad para ordenar horarios de rutas de transporte	PA_77 Verificar adaptación de lista de objetos de horarios y rutas a lista adaptada de listview para presentación en interfaz	4s
	PA_78 Verificar que lista de objetos se ordenen ascendentemente de acuerdo al horario	9s
	PA_79 Verificar que la extracción de la hora actual sea correcta	3s
TI_03 Implementar funcionalidad para extraer próxima salida	PA_80 Verificar que la hora del día sea PM	2s
	PA_81 Verificar que la hora del día sea AM	3s
	PA_82 Verificar la extracción del día de la semana actual sea correcto	4s
	PA_83 Verificar conversión de día numérico a texto	4s
Total	7 pruebas funcionales	29s

Realizado por: Wilmer B. 2018.

Las pruebas funcionales correspondientes al sprint 05, tienen un total de 17 pruebas de aceptación aplicadas y distribuidas en 2 historias de usuario, estas tienen un tiempo total de 66 segundos en haber sido procesadas y aprobadas.

Síntesis de pruebas de aceptación sprint 6

Tabla 7-3: Resultado de tiempo aplicado en pruebas de aceptación en la HU_11.

Historia de usuario: HU_11 Implementación de funcionalidad para mostrar horarios por días		
Tarea de ingeniería	**Prueba de aceptación**	**Tiempo**
TI_01 Convertir días numéricos a días en forma de texto inglés y español	PA_90 Verificar que un día numérico sea convertido a abreviación de día en texto tanto en español como en inglés	4s

TI_02 Creación de funcionalidad para agrupación de días laborables de una agencia de una cooperativa de transporte	PA_91 Verificar la presentación de días laborables concatenados a partir de un conjunto de días numéricos.	6s
Total	2 pruebas funcionales	10s

Realizado por: Wilmer B. 2018.

Las pruebas funcionales correspondientes al sprint 6, tienen un total de 2 pruebas de aceptación aplicadas en una sola historia de usuario, las mismas que tienen un tiempo total de 10 segundos en haber sido procesadas y aprobadas.

Síntesis de pruebas de aceptación sprint 7

Tabla 8-3: Resultado de tiempo aplicado en pruebas de aceptación en la HU_12.

Historia de usuario: HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android		
Tarea de ingeniería	**Prueba de aceptación**	**Tiempo**
TI_01 Implementación de funcionalidad para usar el servicio de localización a través del dispositivo GPS	PA_92 Verificar activación de servicio para localización	3s
	PA_93 Verificar que el proveedor de GPS esté activo	3s
	PA_94 Verificar que el proveedor de GPS esté inactivo	4s
	PA_95 Verificar omisión de altitud en localización	3s
	PA_96 Verificar consumo de energía bajo activo	3s
	PA_97 Verificar si existe proveedor de localización activo	4s
	PA_98 Verificar cuando no existe proveedor de localización activo	2s
	PA_99 Verificar desactivación de servicio de localización	3s
	PA_100 Verificar que el mejor proveedor de localización no sea el de red	4s

	PA_101 Verificar que el proveedor de localización esté en modo pasivo	3s
	PA_102 Verificar que el mejor proveedor de localización sea el GPS	4s
TI_02 Implementación de funcionalidad para usar el servicio de localización a través de la red celular a la que se encuentra conectado	PA_103 Verificar que el proveedor de localización de red esté activo	3s
	PA_104 Verificar que el proveedor de localización de red esté inactivo	3s
	PA_105 Verificar que el mejor proveedor de localización no sea GPS	3s
	PA_106 Verificar que el mejor proveedor de localización GPS esté inactivo	3s
Total	15 pruebas funcionales	48s

Realizado por: Wilmer B. 2018.

Tabla 9-3: Resultado de tiempo aplicado en pruebas de aceptación en la HU_13.

Historia de usuario: HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte		
Tarea de ingeniería	**Prueba de aceptación**	**Tiempo**
TI_01 Implementar funcionalidad para verificar conexión a una red wifi, o por datos móviles	PA_107 Verificar si existe una conexión a internet activa	3s
	PA_108 Verificar que haya conexión a red activa wifi	5s
	PA_109 Verificar cuando no hay conexión a red wifi activa	4s
	PA_110 Verificar cuando hay conexión a red activa con datos móviles	5s
	PA_111 Verificar cuando no hay conexión a red activa con datos móviles	4s
	PA_112 Verificar activación de servicio para acceder a internet	3s
	PA_113 Verificar cuando no existe servicio activo para acceso a internet	3s
	PA_114 Verificar cuando existe conectividad y disponibilidad de red con datos móviles	3s
TI_02 Crear funcionalidad para consumo de servicios web mediante método post	PA_115 Verificar si se abrió una petición http de consumo de servicio web exitoso	3s
	PA_116 Verificar si el estado de petición fue aceptado	3s
	PA_117 Verificar si la petición fue rechazada	3s
	PA_118 Verificar que el resultado devuelto a petición no sea vacío	8s
	PA_119 Verificar cuando no hay resultado de servicio web	7s

TI_03 Crear funcionalidad para extracción y procesamiento de resultado devuelto por servicio web en hosting	PA_120 Verificar la disponibilidad de asientos desde respuesta de servicio web	27s
	PA_121 Verificar que el resultado del servicio web sea una consulta correcta	3s
	PA_122 Verificar que el resultado del servicio web sea una consulta incorrecta	4s
	PA_123 Verificar extracción de número de asientos disponibles	63s
	PA_124 Verificar extracción de asientos ocupados sea correcto	3s
TI_04 Creación de funcionalidad para ejecutar petición de boletos disponibles en segundo plano	PA_125 Verificar que ejecución de consumo de servicio web en segundo plano.	7s
Total	19 pruebas funcionales	161s

Realizado por: Wilmer B. 2018.

Las pruebas funcionales correspondientes al sprint 07, tienen un total de 34 pruebas de aceptación aplicadas y distribuidas en 2 historias de usuario, estas tienen un tiempo total de 209 segundos en haber sido procesadas y aprobadas.

Síntesis de pruebas de aceptación sprint 8

Tabla 10-3: Resultado de tiempo aplicado en pruebas de aceptación en la HU_14.

Historia de usuario: HU_14 Implementación de funcionalidad favoritos		
Tarea de ingeniería	**Prueba de aceptación**	**Tiempo**
TI_01 Implementar funcionalidad para extraer lista de favoritos	PA_126 Verificar que se cargue lista de rutas favoritas desde la base de datos local	4s
	PA_127 Verificar cuando la lista de rutas favoritas esté vacía	3s
TI_02 Implementar funcionalidad parar extraer lista de objetos favoritos a lista de cadena de string	PA_128 Verificar formateo de lista de objetos con rutas favoritas a lista de strings para presentación de estos.	16s

TI_03 Implementar funcionalidad para agregar una ruta favorita a la lista	PA_129 Verificar si dicha ruta favorita ya existe	3s
	PA_130 Verificar si una ruta fue agregada correctamente	3s
	PA_131 Verificar si una ruta específica que se encuentra dentro de favoritos	3s
TI_04 Implementar funcionalidad para extraer detalles de una ruta favorita en base a su id	PA_132 Verificar extracción de una ruta favorita en base a su id y hora	4s
Total	7 pruebas funcionales	36s

Realizado por: Wilmer B. 2018.

Tabla 11-3: Resultado de tiempo aplicado en pruebas de aceptación en la HU_15.

Historia de usuario: HU_15 Implementación de funcionalidad llamada telefónica y cargar datos de agencia de cooperativa		
Tarea de ingeniería	**Prueba de aceptación**	**Tiempo**
TI_01 Implementar funcionalidad para extraer datos de una agencia de transporte	PA_133 Verificar extracción de datos de una agencia de cooperativa	14s
	PA_134 Verificar cuando no existe agencia o datos de agencia de una cooperativa de transporte	3s
TI_02 Implementar funcionalidad para realizar una llamada telefónica desde la aplicación	PA_135 Verificar que permiso de llamada se encuentre activo	3s
	PA_136 Verificar cuando no existe permiso de llamada activo	3s
	PA_137 Verificar conversión de número en formato texto a formato numérico para llamar	7s
Total	5 pruebas funcionales	30s

Realizado por: Wilmer B. 2018.

Las pruebas funcionales correspondientes al sprint 08, tienen un total de 12 pruebas de aceptación aplicadas y distribuidas en 2 historias de usuario, estas tienen un tiempo total de 66 segundos en haber sido procesadas y aprobadas.

Síntesis de pruebas de aceptación sprint 9

Tabla 12-3: Resultado de tiempo aplicado en pruebas de aceptación en la HU_17.

Historia de usuario: HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting		
Tarea de ingeniería	**Prueba de aceptación**	**Tiempo**
TI_01 Implementar funcionalidad para verificar que el dispositivo no esté conectado a datos móviles	PA_44 Verificar activación de servicio de wifi	3s
	PA_45 Verificar conexión a la red activa	3s
	PA_46 Verificar cuando no hay conexión a la red activa	3s
	PA_47 Verificar que el servicio de conexión a la red esté disponible	3s
	PA_48 Verificar que el tipo de conexión sea wifi	4s
	PA_49 Verificar que el tipo de conexión de red sea datos móviles	4s
TI_02 Implementar funcionalidad para realizar petición de consumo de servicio web de hosting para sincronización de base de datos	PA_150 Verificar que la petición http sea aceptada.	3s
TI_03 Implementar funcionalidad para extracción de resultados de petición de sincronización.	PA_151 Verificar que la respuesta del servicio web haya sido exitosa	5s
	PA_152 Verificar que la respuesta del servicio web haya sido fallida	4s
TI_04 Implementar funcionalidad para la eliminación de contenido y tablas anteriores de la base de datos local	PA_153 Verificar la existencia de las tablas: COOPERATIVA, DEPENDENCIA, TERMINAL, RUTA, HORARIO, TERM_PERTENECE_DEPEN, DEPEN_POSEE_RUTAS, PROVINCIAS, CIUDADES, FAVORITO de la base de datos local.	10x3 =30s
	PA_154 Verificar la eliminación de contenido de todas las tablas de la base de datos local.	4s
TI_06 Implementar funcionalidad para inserción de datos de tablas	PA_164 Verificar la inserción correcta de los datos de la tabla COOPERATIVA	4s
	PA_165 Verificar la inserción correcta de los datos de la tabla DEPENDENCIA	4s
	PA_166 Verificar la inserción correcta de los datos de la tabla TERMINAL	3s

locales con la información extraída desde la respuesta del servicio web	PA_167 Verificar la inserción correcta de los datos de la tabla RUTA	7s
	PA_168 Verificar la inserción correcta de los datos de la tabla HORARIO	8s
	PA_169 Verificar la inserción correcta de los datos de la tabla TERM_PERTENECE_DEPEN	4s
	PA_170 Verificar la inserción correcta de los datos de la tabla DEPEN_POSEE_RUTAS	4s
	PA_171 Verificar la inserción correcta de los datos de la tabla PROVINCIAS	3s
	PA_172 Verificar la inserción correcta de los datos de la tabla CIUDADES	3s
TI_07 Implementar funcionalidad para ejecución de sincronización en segundo plano	PA_173 Verificar que el proceso de sincronización de base de datos se lleve a cabo en segundo plano.	7s
Total	30 pruebas funcionales	113s

Realizado por: Wilmer B. 2018.

Las pruebas funcionales correspondientes al sprint 9, tienen un total de 30 pruebas de aceptación aplicadas en una sola historia de usuario, las mismas que tienen un tiempo total de 113 segundos en haber sido procesadas y aprobadas.

En la presente síntesis existe un detalle en la TI_04 (Tarea de ingeniería 04) junto a la PA_153 (Prueba de aceptación) en este proceso se ha optado por documentar en una sola prueba 10 pruebas de aceptación, esto considerando que es un proceso es repetitivo, la verificación de cada una de las tablas de la base de datos local y a la cual se ha obtenido un tiempo de 3 segundos por cada una de las verificaciones.

Síntesis del proyecto

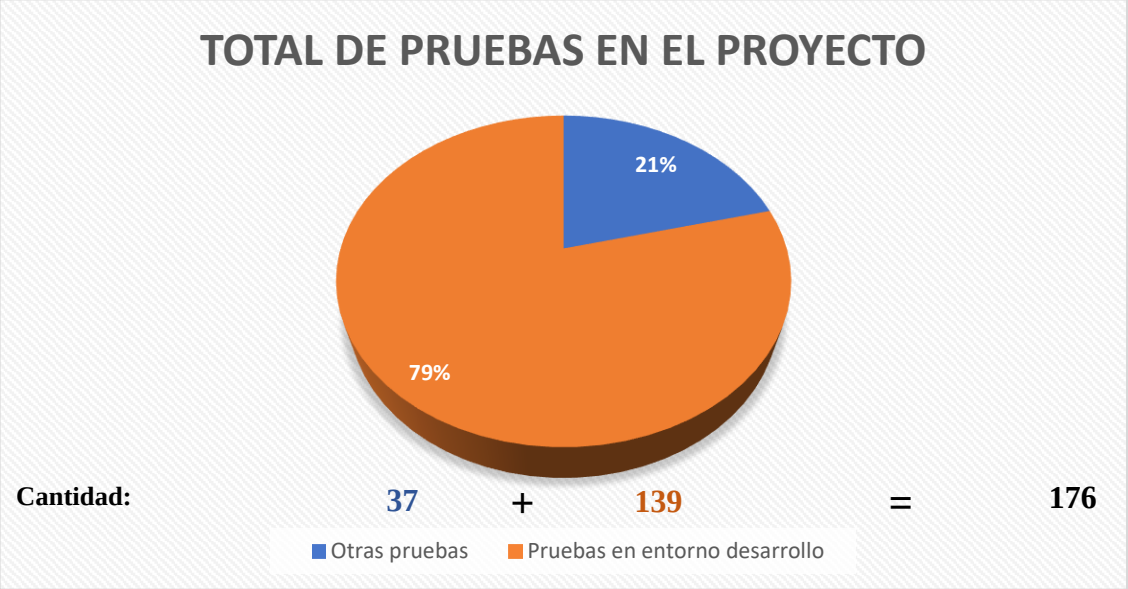


Gráfico 2-3: Distribución de pruebas de aceptación en el proyecto.

Realizado por: Wilmer B. 2018.

Tabla 13-3: Cantidad de pruebas dentro del proyecto.

Pruebas	Cantidad
Otras (documentación, interface, etc.)	37
En entorno de desarrollo	139
Total	176

Realizado por: Wilmer B. 2018.

Para la presentación de resultados se ha divido las pruebas en dos partes, con un total de 176 pruebas de aceptación siguiendo la metodología scrum, se ha tomado una parte en la que se encuentran pruebas varias como lo son pruebas de interfaz y pruebas de documentación (diseño de base de datos, arquitectura, gestión de riesgos, etc.) con un número de pruebas igual a 37 y correspondiendo al 21% del total de pruebas dentro del proyecto, estas no están dentro de los objetivos de análisis de este proyecto por lo cual serán descartadas. La otra parte de pruebas corresponde a pruebas unitarias que evalúan la funcionalidad del sistema, las cuales serán analizadas para la presentación de resultados, estas pruebas tienen un número igual a 139 que corresponde al 79% del total de pruebas aplicadas en este proyecto.

Estas pruebas unitarias funcionales han sido implementadas y a la vez se ha tomado en cuenta el tiempo que se demora en el paso de la verificación del resultado esperado, este tiempo se

encuentra registrado en cada una de las tablas de pruebas de aceptación. Entonces para esto se ha tomado en cuenta 8 sprints que van desde el sprint 2 al 9 y con un total de 12 Historias de usuario.

Análisis de pruebas que evalúan la funcionalidad del sistema

A continuación, se mostrará el tiempo registrados dentro de cada sprint por el conjunto de pruebas que pertenece a este, así como también se reflejará la distribución de las pruebas según el avance de sprints.

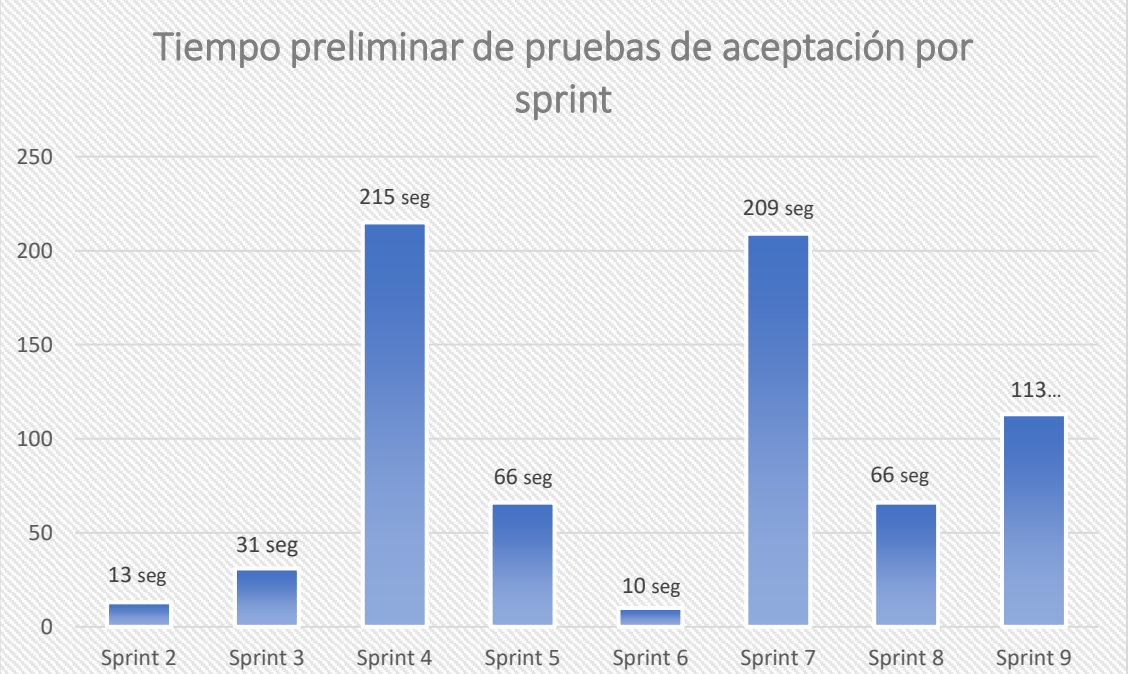


Gráfico 3-3: Resultado de tiempos de evaluación de pruebas de aceptación por sprint.

Realizado por: Wilmer B. 2018.

Tabla 14-3: Tiempo preliminar en pruebas manuales.

Sprints	2	3	4	5	6	7	8	9	**7 sprints**
Tiempo	13seg	31seg	215seg	66seg	10seg	209seg	66seg	113seg	**723seg**

Realizado por: Wilmer B. 2018.

En el **gráfico 3-3** se puede apreciar el resultado preliminar el cual consta de los tiempos por sprint, correspondientes únicamente a la parte de verificación del resultado que se espera en dicha prueba, dicho de otra manera, esta parte es el tiempo que se demora el evaluador en verificar visualmente los resultados que presenta el sistema.

Este resultado con tiempos preliminares es una pequeña parte del total de tiempo que toma hacer una prueba de aceptación, ya que, en el proceso de realización y aprobación de esta se debe tomar en cuenta varios factores como lo son:

- Tiempo que se tarda en encender el emulador cada vez que se vaya a evaluar el sistema
- Tiempo que se demora el sistema en ejecutar la última versión del sistema que se quiere evaluar
- Y por último el tiempo tomado en seguir el proceso de documentación de cada una de las pruebas de aceptación.

Entonces con un total de 139 pruebas de aceptación que evalúan la funcionalidad del sistema y realizadas siguiendo la metodología de desarrollo ágil scrum se procede a mostrar la cantidad de estas pruebas desarrollada por cada uno de los sprints.

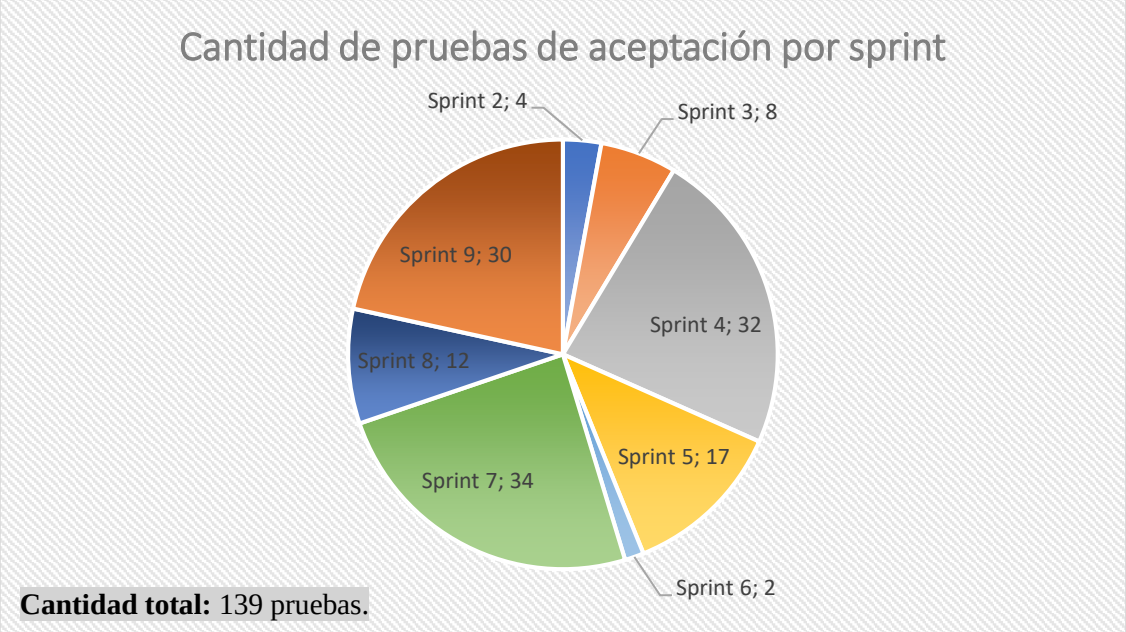


Gráfico 4-3: Cantidad de pruebas de aceptación por sprint.

Realizado por: Wilmer B. 2018.

Tabla 15-3: Cantidad de pruebas por sprint.

Sprints	2	3	4	5	6	7	8	9	**7 sprints**
# Pruebas	4	8	32	17	2	34	12	30	**139**

Realizado por: Wilmer B. 2018.

En el **gráfico 4-3** se puede apreciar la distribución de un número determinado de pruebas por sprint, las cuales se han ido desarrollado siguiendo la planificación planteada llegando a tener los presentes resultados que contribuirán a un análisis de resultados finales.

Resultados finales en pruebas de aceptación

Para el resultado final se ha tenido en cuenta 4 variables que entran en juego al momento de tomar tiempos para evaluar las pruebas de aceptación, estas se detallan a continuación:

A. **Tiempo que se tarda en encender el emulador:** este tiempo puede variar dependiendo de las características de una computadora, en este caso se ha obtenido un tiempo de 1 minuto con 52 segundos en el cual se enciende el emulador. Para obtener la cantidad total de veces que se enciende el emulador se debe verificar las fechas de cada una de las pruebas de aceptación con esto se obtendrá el número exacto de veces que se encendió el emulador.

B. **Tiempo que tarda el emulador en ejecutar la última versión del sistema:** este tiempo de demora se presentará cada vez que se evalúe una prueba de aceptación, en nuestro caso se ha obtenido el tiempo de 13 segundos que tarda el sistema en ejecutar su última versión. En cuanto a la cantidad de veces que se presentan estos tiempos es igual al número de pruebas de aceptación.

C. **Tiempo que se tarda en documentar una prueba:** Este tiempo se presenta en cada una de las pruebas de aceptación, existe una gran variabilidad en la toma de estos tiempos ya que depende de la persona que documente, ya sea que escriba de manera rápida o no esto puede variar, así que para nuestro proyecto se ha tomado en cuenta un tiempo medio de 8 minutos con 45 segundos, esto debido a que ya antes hemos definido el formato que llevará estas pruebas de aceptación.

D. **Tiempo que se tarda en verificar el resultado esperado:** Al momento de llevar a cabo el paso a paso de la documentación de la prueba existe un punto donde se verifica el resultado presentado por el sistema con el resultado esperado en la documentación de esta manera se podrá verificar si dicha prueba fue aceptada o no. Dicho tiempo se encuentra registrado en cada una de las tablas de documentación de prueba de aceptación.

A continuación, se obtendrá la cantidad total de veces que se encendió el emulador, así como dentro de cada sprint.

Tabla 16-3: Cantidad de veces en encender el emulador.

Sprint	Prueba Aceptación	Fecha	Cantidad
2	PA_07	17/10/17	4
	PA_10	18/10/17	
	PA_12	20/10/17	

	PA_14	25/10/17	
3	PA_22 – PA_28	06/11/17	2
	PA_29	08/11/17	
4	PA_30 – PA_31	13/11/17	7
	PA_32 – PA_40	14/11/17	
	PA_41 – PA_49	15/11/17	
	PA_51 – PA_54	17/11/17	
	PA_55	20/11/17	
	PA_56 – PA_60	21/11/17	
	PA_61 – PA_62	22/11/17	
5	PA_65 – PA_66	29/11/17	5
	PA_67 – PA_68	30/11/17	
	PA_69 – PA_70	01/12/17	
	PA_71 – PA_74	04/12/17	
	PA_77 – PA_83	07/12/17	
6	PA_90	19/12/17	2
	PA_91	21/12/17	
7	PA_92 – PA_102	27/12/17	6
	PA_103 – PA_106	28/12/17	
	PA_107 – PA_114	29/12/17	
	PA_115 – PA_119	03/01/18	
	PA_120 – PA_124	04/01/18	
	PA_125	05/01/18	
8	PA_126 – PA_127	10/01/18	7
	PA_128	11/01/18	
	PA_129 – PA_131	12/01/18	
	PA_132	16/01/18	
	PA_133 – PA_134	18/01/18	
	PA_135	19/01/18	
	PA_136 -– PA_137	19/01/18	
9	PA_144-149	25/01/18	6
	PA_150	26/01/18	
	PA_151 – PA_152	29/01/18	
	PA_153 – PA_154	30/01/18	
	PA_164 – PA_172	31/01/18	
	PA_173	01/02/18	
		Cantidad total	39 veces

Realizado por: Wilmer B. 2018.

Entonces:

Variable A = 39 veces.

Tiempo de encendido del emulador: 1 minuto con 52 segundos o 112 segundos

Tota variable A = Cantidad de veces que se enciende el emulador * Tiempo de encendido
Tiempo total de variable A = 39 * 112 = **4368 segundos /72min con 48s/1h con 12min y 48s.**

Ahora, para encontrar la variable B obtendremos:
Tiempo en ejecutar la última versión del sistema: 13 segundos
Variable B = Total de pruebas de aceptación funcionales * Tiempo en ejecutarse la última versión.
Tiempo total de variable B = 139 * 13 = **1807 segundos /30 minutos con 7 segundos**

Variable C, ahora se encontrará el tiempo total tomado en la documentación de las pruebas:
Tiempo en proceso de documentación: 8 minutos con 45 segundos o 525 segundos
Variable C = Total de pruebas de aceptación funcionales * Tiempo de proceso de documentación
Tiempo total de variable C = 139 * 525 = **72975 segundos/1216**min **con 15s/20h con 16min y 15s.**

Ahora para encontrar el tiempo total de la variable D, haremos referencia a los datos presentados en el **gráfico 3-3**, ya que en este se encuentra detallado los tiempos de evaluación de pruebas en el punto de verificación de resultado esperado, los mismo que están presentados por cada sprint.

Tabla 17-3: Resumen tiempos en pruebas de aceptación por sprint.

Sprint	2	3	4	5	6	7	8	9	**Total**
Tiempo (segundos)	13	31	215	66	10	209	66	113	**723s/12min 3s**

Realizado por: Wilmer B. 2018.

Entones el tiempo total que se ha tomado en realizar las pruebas de aceptación funcionales de manera manual o siguiendo la metodología scrum será la suma de las 4 variables antes definidas:
Tiempo total empleado en pruebas de aceptación: A + B + C + D

Tiempo total = 4368s + 1807s + 72975s + 723s = 79873 segundos/1331min con 13s/22h con 10min y 13s.

La razón por la cual se ha separado el tiempo aplicado en pruebas en cuatro variables es porque tanto en las variables A y B los tiempos son constantes en todo el proyecto, mientras que la variable C es parcialmente contante ya que este depende en parte de la habilidad humana para narrar y escribir; por otro lado, la variable D es la que con más frecuencia varía en cada una de las pruebas de aceptación aplicadas, esto debido a que este paso depende completamente del personal humano la capacidad que tiene para observar el resultado esperado y el resultado

arrojado por el sistema, así como también el criterio con el que decide aprobar o dar por fallida una u otra prueba de aceptación.

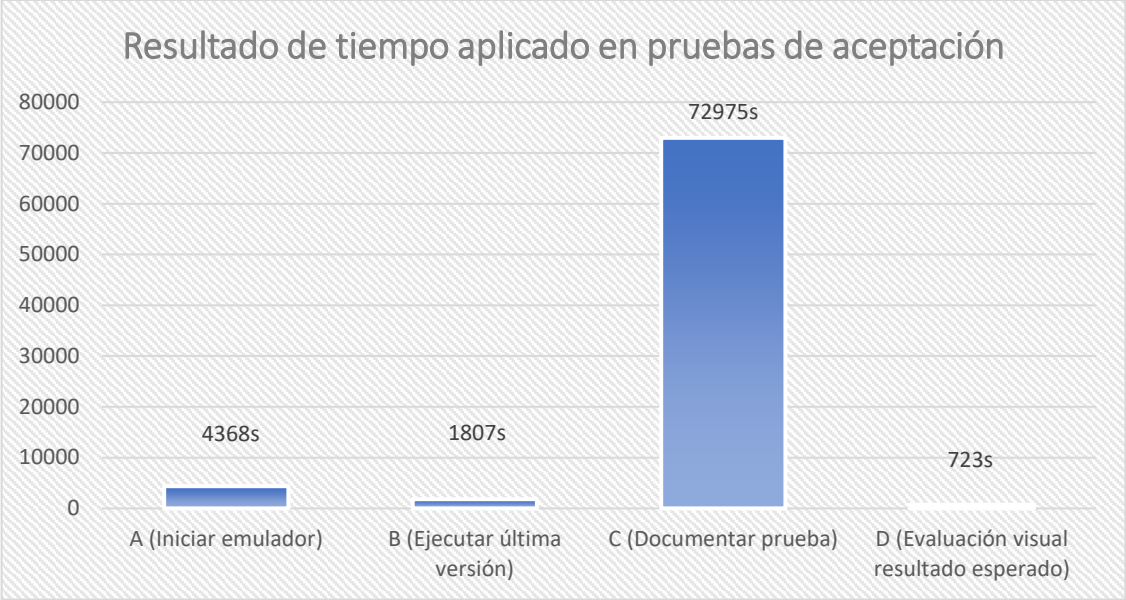


Gráfico 5-3: Resultado final tiempos en pruebas de aceptación.

Realizado por: Wilmer B. 2018.

Tabla 18-3: Variables de tiempo en pruebas manuales.

Variables	Tiempo
A (Iniciar emulador)	4368seg
B (Ejecutar última versión)	1807seg
C (Documentar prueba)	72975seg
D (Evaluación visual resultado esperado)	723seg

Realizado por: Wilmer B. 2018.

3.3 Resultados de pruebas automáticas

De igual manera como ya antes se ha presentado los resultados de pruebas de aceptación ahora se procederá a presentar cada uno de los resultados de los test unitarios automatizados.

A continuación, se muestra los tiempos de ejecución obtenidos en cada una de las pruebas unitarias, de igual manera están separadas por sprint y por último se muestra un resultado total o general de estos resultados obtenidos. Los tiempos recolectados se han tomado en cuenta en las unidades de tiempo igual a **s:** segundos y **ms:** milisegundos, estos tiempos se encuentran respaldados en un conjunto de imágenes que se encuentran en el **Anexo D**.

Síntesis de pruebas unitarias automatizadas sprint 2

Tabla 19-3: Resultado de tiempo en pruebas automatizadas en la HU_01.

Historia de usuario: HU_01 Codificación de scripts SQL para transacciones en base de datos		
Tarea de ingeniería	**Caso de prueba**	**Tiempo**
TI_04 Implementación de funcionalidad para crear base de datos y conexión a esta.	TC_01 Verificar que la base de datos local haya sido creada	0ms
TI_03 Implementación de script para eliminación de tablas	TC_02 Verificar eliminación de tabla COOPERATIVA	25ms
	TC_03 Verificar eliminación de tabla DEPENDENCIA	25ms
	TC_04 Verificar eliminación de tabla DEPENDENCIA POSEE RUTAS	25ms
	TC_05 Verificar eliminación de tabla HORARIO	26ms
	TC_06 Verificar eliminación de tabla RUTA	50ms
	TC_07 Verificar eliminación de tabla TERMINAL	25ms
	TC_08 Verificar eliminación de tabla TERMINAL PERTENCE DEPENDENCIA	25ms
	TC_09 Verificar eliminación de tabla PROVINCIAS	0ms
	TC_10 Verificar eliminación de tabla CIUDADES	25ms
	TC_11 Verificar eliminación de tabla FAVORITO	50ms
TI_02 Implementación de script para eliminación de contenido de tablas	TC_12 Verificar eliminación del contenido existente en tabla COOPERATIVA en base de datos local	1s 357ms
	TC_13 Verificar eliminación del contenido existente en tabla DEPENDENCIA en base de datos local	1s 589ms
	TC_14 Verificar eliminación del contenido existente en tabla DEPENDENCIA POSEE RUTAS en base de datos local	1s 332ms
	TC_15 Verificar eliminación del contenido existente en tabla HORARIO en base de datos local	1s 381ms

	TC_16 Verificar eliminación del contenido existente en tabla RUTA en base de datos local	1s 335ms
	TC_17 Verificar eliminación del contenido existente en tabla TERMINAL en base de datos local	1s 458ms
	TC_18 Verificar eliminación del contenido existente en tabla TERMINAL PERTENCE DEPENDENCIA en base de datos local	1s 532ms
	TC_19 Verificar eliminación del contenido existente en tabla PROVINCIAS en base de datos local	1s 382ms
	TC_20 Verificar eliminación del contenido existente en tabla CIUDADES en base de datos local	1s 484ms
	TC_21 Verificar eliminación del contenido existente en tabla FAVORITO en base de datos local	1s 506ms
TI_01 Implementación de script para creación de tablas	TC_22 Verificar la creación de tabla COOPERATIVA en base de datos local	50ms
	TC_23 Verificar la creación de tabla DEPENDENCIA en base de datos local	25ms
	TC_24 Verificar la creación de tabla DEPENDENCIA POSEE RUTAS en base de datos local	25ms
	TC_25 Verificar la creación de tabla HORARIO en base de datos local	51ms
	TC_26 Verificar la creación de tabla RUTA en base de datos local	50ms
	TC_27 Verificar la creación de tabla TERMINAL en base de datos local	75ms
	TC_28 Verificar la creación de tabla TERMINAL PERTENCE DEPENDENCIA en base de datos local	75ms
	TC_29 Verificar la creación de tabla PROVINCIAS en base de datos local	51ms
	TC_30 Verificar la creación de tabla CIUDADES en base de datos local	75ms
	TC_31 Verificar la creación de tabla FAVORITO en base de datos local	51ms
Total	31 pruebas unitarias funcionales	16s 916ms

Realizado por: Wilmer B. 2018.

Las pruebas funcionales correspondientes al sprint 2, tienen un total de 31 pruebas unitarias automatizadas en una sola historia de usuario, las mismas que tienen un tiempo total de 16 segundos con 916 milisegundos en haber sido procesadas y proporcionar un resultado de aprobación o falla de estas.

Síntesis de pruebas unitarias automatizadas sprint 3

Tabla 20-3: Resultado de tiempo en pruebas automatizadas en la HU_03.

Historia de usuario: HU_03 Implementación de servicios web Geocoding API de Google Maps a través de petición GET		
Tarea de ingeniería	**Caso de prueba**	**Tiempo**
TI_01 Implementación de funcionalidad para realizar peticiones de consumo de web service a través de método get	TC_01 Verificar que haya respuesta del consumo de servicio web de Google Maps	708ms
	TC_02 Chequear cuando el consumo de servicio web geocoding no fue exitoso	1s 188ms
	TC_03 Chequear cuando consumo de servicio web geocoding fue exitosa pero no obtuvo resultados	2s 25ms
	TC_04 Verificar consumo de servicio web geocoding sea exitoso	634ms
	TC_05 Verificar que la respuesta del consumo de servicio web geocoding obtuvo una lista de resultados	2s 780ms
	TC_06 Verificar que la conexión http es aceptada	456ms
	TC_07 Chequear que la conexión htttp fue rechazada	0ms
Total	7 pruebas unitarias funcionales	7s 791ms

Realizado por: Wilmer B. 2018.

Las pruebas funcionales correspondientes al sprint 3, tienen un total de 7 pruebas unitarias automatizadas en una sola historia de usuario, las mismas que tienen un tiempo total de 7 segundos con 791 milisegundos en haber sido procesadas y proporcionar un resultado de aprobación o falla de estas.

Síntesis de pruebas unitarias automatizadas sprint 4

Tabla 21-3: Resultado de tiempo en pruebas automatizadas en la HU_04.

Historia de usuario: HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps		
Tarea de ingeniería	**Caso de prueba**	**Tiempo**
TI_01 Implementación de funcionalidad para recolectar resultados de petición a servicio web geocoding	TC_01 Chequear que la lista de direcciones filtradas esté vacía	0ms
TI_02 Implementación de funcionalidad para filtrar resultado devuelvo de servicio web geocoding.	TC_02 Verificar que existe lista de direcciones filtradas	1s 35ms
TI_03 Implementación de funcionalidad para formatear cadenas de direcciones.	TC_03 Formatear lista de direcciones filtradas	25ms
TI_04 Implementación de funcionalidad para encontrar coincidencias de provincias del Ecuador con el listado de resultados devuelto por el servicio geocoding.	TC_04 Cargar provincias desde base de datos local en una lista	25ms
	TC_05 Chequear que la lista de provincias de base de datos local este vacía	1s 532ms
	TC_06 Encontrar coincidencia de provincia entre la lista de direcciones y la lista de provincias extraídas de la base de datos local	0ms
	TC_07 Chequear cuando no se encuentra coincidencia de provincia entre la lista de direcciones y la lista de provincias extraídas de la base de datos local	0ms
TI_05 Implementación de funcionalidad para encontrar coincidencias de ciudades del	TC_08 Cargar lista de ciudades desde base de datos local en una lista, en base al id de una provincia	25ms
	TC_09 Chequear cuando la lista de ciudades extraídas de la base en base al id de una provincia sea vacía	1s 330ms
	TC_10 Encontrar coincidencia de ciudad entre la lista de direcciones y la lista de ciudades extraídas de la base de datos	0ms

Ecuador con el listado de resultados devuelto por el servicio geocoding.	TC_11 Chequear cuando no se encontraron coincidencias de ciudad entre la lista de direcciones y la lista de ciudades extraídas de la base de datos	25ms
TI_06 Implementación de funcionalidad para obtener la distancia entre dos coordenadas geográficas.	TC_12 Obtener distancia entre dos coordenadas gps a través de geometría esférica	0ms
	TC_13 Transformar cantidad matemática normal a radianes	26ms
TI_07 Implementar funcionalidad para obtener ciudad más cercana de acuerdo a una dirección específica	TC_14 Obtener el índice del número menor de una lista de distancias entre ciudades	26ms
	TC_15 Cargar todas las ciudades desde base de datos local en una lista	0ms
	TC_16 Chequear cuando la lista de todas las ciudades extraída de la base de datos local esté vacía	1s 283ms
	TC_17 Verificar ciudad cercana en base a otra	25ms
	TC_18 Chequear cuando no existe ciudad cercana en base a otra	25ms
	TC_19 Chequear cuando si existe ciudad cercana en base a otra	25ms
	TC_20 Verificar la existencia de una ciudad específica dentro de la base de datos en base a una ciudad ingresada	26ms
	TC_21 Verificar si no existen ciudad origen, en base a una ciudad ingresada	0ms
Total	21 pruebas unitarias funcionales	5s 433ms

Realizado por: Wilmer B. 2018.

Tabla 22-3: Resultado de tiempo en pruebas automatizadas en la HU_06.

Historia de usuario: HU_06 Implementación de funcionalidad para consulta de horarios por transporte		
Tarea de ingeniería	**Caso de prueba**	**Tiempo**
TI_01 Crear funcionalidad para extracción de lista de cooperativas de transporte	TC_01 Cargar lista de cooperativas de transporte desde la base de datos local	1s 508ms
	TC_02 Chequear cuando la lista de transportes esté vacía, cuando no se encuentre ninguna cooperativa de transporte	1s 462ms
TI_03 Crear funcionalidad para formatear información de rutas almacenadas en lista de objetos pasando a lista de strings	TC_03 Convertir lista de objetos de cooperativas de transporte a lista de strings	1s 206ms
TI_02 Crear funcionalidad para extracción de lista de rutas que pertenecen a una cooperativa de transporte	TC_04 Cargar lista de rutas en base al id de una cooperativa de transporte desde la base de datos local	1s 157ms
	TC_05 Chequear cuando la lista de rutas en base al id de una cooperativa de transportes esté vacía	1s 434ms
TI_04 Crear funcionalidad para obtener lista de dependencias que pertenecen a una cooperativa de transporte	TC_06 Extraer lista de rutas de lista de objetos dependencia a lista de strings ordenadas alfabéticamente	1s 333ms
	TC_07 Obtener id de dependencia posee ruta en base a ciudad origen y destino de una cooperativa de transporte	1s 7ms
	TC_08 No se ha obtenido id de dependencia posee ruta en base a ciudad origen y destino de una cooperativa de transporte	1s 81ms
	TC_09 Obtener las dependencias o agencia junto con las rutas, en base al id de la misma	1s 108ms
	TC_10 Verificar que no existe información de dependencia o agencia que poseen rutas, en base a su id	859ms
TI_05 Crear funcionalidad para extracción de lista de horarios en base a una ruta de una agencia de cooperativa de transporte específica.	TC_11 Obtener lista de horarios de una ruta en base al id de Dependencia posee ruta	1s 409ms
	TC_12 Verificar que no existan datos registrados de los horarios de una ruta en base al id de dependencia posee ruta	1s 132ms
Total	12 pruebas unitarias funcionales	14s 693ms

Realizado por: Wilmer B. 2018.

Las pruebas funcionales correspondientes al sprint 04, tienen un total de 33 pruebas unitarias automatizadas y se encuentran distribuidas en 2 historias de usuario, estas tienen un tiempo total de 20 segundos con 126 milisegundos en haber sido procesadas y proporcionar un resultado de aprobación o falla de estas.

Síntesis de pruebas unitarias automatizadas sprint 5

Tabla 23-3: Resultado de tiempo en pruebas automatizadas en la HU_08.

Historia de usuario: HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino		
Tarea de ingeniería	**Caso de prueba**	**Tiempo**
TI_01 Creación de funcionalidad para extraer lista de ciudades origen desde la base de datos	TC_01 Cargar lista ciudades origen desde base de datos local	1s 105ms
	TC_02 Chequear cuando no hay lista de ciudades origen desde base de datos local vacía	1s 209ms
TI_02 Creación de funcionalidad para extraer lista de ciudades destino basado en una ciudad origen	TC_03 Cargar lista ciudades destino desde la base de datos local	1s 285ms
	TC_04 Chequear cuando la lista ciudades destino está vacía	1s 155ms
TI_03 Crear funcionalidad para formatear lista de objetos ruta a lista de strings	TC_05 Extraer lista de Strings de ciudades destino de una lista de objetos de ruta	984ms
	TC_06 Formatear cadena de texto a formato donde las palabras empiezan con letra mayúscula	1s 331ms
TI_04 Crear funcionalidad para extraer lista de horarios en base a una ruta, sin importar la cooperativa de transporte	TC_07 Obtener lista de horarios en base a una ruta específica	804ms
	TC_08 Chequear que la lista de horarios en base a una ruta específica este vacía	1s 331ms
	TC_09 Obtener id de una ruta	829ms
	TC_10 Chequear cuando no se ha obtenido id de ruta	1s 31ms
Total	10 pruebas unitarias funcionales	11s 64ms

Realizado por: Wilmer B. 2018.

Tabla 24-3: Resultado de tiempo en pruebas automatizadas en la HU_09.

Historia de usuario: HU_09 Implementación de funcionalidad para obtener la próxima salida en base a la hora actual		
Tarea de ingeniería	**Caso de prueba**	**Tiempo**
TC_84 Verificar la hora actual es AM	TC_01 Adaptar lista de información de horarios, rutas y transportes para presentación en interfaz	1s 908ms
	TC_02 Ordenar ascendente la lista de horarios de una ruta específica adaptada	1s 131ms
	TC_07 Obtener hora actual normal	1s 154ms
TI_03 Implementar funcionalidad para extraer próxima salida	TC_03 Verificar la hora actual es AM	1s 761ms
	TC_04 Verificar la hora actual es PM	1s 330ms
	TC_05 Obtener el día de la semana actual numéricamente	1s 310ms
	TC_06 Convertir día numérico actual a día actual escrito	1s 156ms
Total	7 pruebas unitarias funcionales	2s 550ms

Realizado por: Wilmer B. 2018.

Las pruebas funcionales correspondientes al sprint 05, tienen un total de 17 pruebas unitarias automatizadas y se encuentran distribuidas en 2 historias de usuario, estas tienen un tiempo total de 13 segundos con 614 milisegundos en haber sido procesadas y proporcionar un resultado de aprobación o falla de estas.

Síntesis de pruebas unitarias automatizadas sprint 6

Tabla 25-3: Resultado de tiempo en pruebas automatizadas en la HU_11.

Historia de usuario: HU_11 Implementación de funcionalidad para mostrar horarios por días		
Tarea de ingeniería	**Caso de prueba**	**Tiempo**
TI_02 Creación de funcionalidad para agrupación de días laborables de una agencia de una cooperativa de transporte	TC_01 Convertir día numérico a abreviación de día	0ms
	TC_02 Concatenación de días numéricos a días escritos laborables	25ms
Total	2 pruebas unitarias funcionales	25ms

Realizado por: Wilmer B. 2018.

Las pruebas funcionales correspondientes al sprint 6, tienen un total de 2 pruebas unitarias automatizadas en una sola historia de usuario, las mismas que tienen un tiempo total de 25 milisegundos en haber sido procesadas y proporcionar un resultado de aprobación o falla de estas.

Síntesis de pruebas unitarias automatizadas sprint 7

Tabla 26-3: Resultado de tiempo en pruebas automatizadas en la HU_12.

Historia de usuario: HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android		
Tarea de ingeniería	**Caso de prueba**	**Tiempo**
TI_01 Implementación de funcionalidad para usar el servicio de localización a través del dispositivo GPS	TC_01 Activar servicio para localización LocationManager para uso de servicio de localización	0ms
	TC_02 Verificar proveedor de localización gps activo	25ms
	TC_03 Verificar proveedor de localización gps inactivo	0ms
	TC_06 Omitir dato altitud en la localización	0ms

	TC_07 Activar consumo de energía bajo	0ms
	TC_08 Existe proveedor de localización activo	0ms
	TC_09 Chequear cuando no existe proveedor de localización activo	26ms
	TC_10 Desactivar servicio de localización, cuando ya no se necesita	0ms
	TC_11 Chequear que el mejor proveedor de localización no sea el de red	0ms
	TC_12 Verificar que el proveedor de localización esté en modo pasivo	0ms
	TC_13 Verificar que el mejor proveedor de localización sea el dispositivo gps	0ms
TI_02 Implementación de funcionalidad para usar el servicio de localización a través de la red celular a la que se encuentra conectado	TC_04 Verificar proveedor de localización de red activo	25ms
	TC_05 Verificar proveedor de localización de red inactivo	0ms
	TC_14 Chequear que el mejor proveedor de localización no sea el dispositivo gps	0ms
	TC_15 Verificar que el proveedor de localización gps esté inactivo	0ms
Total	15 pruebas unitarias funcionales	76ms

Realizado por: Wilmer B. 2018.

Tabla 27-3: Resultado de tiempo en pruebas automatizadas en la HU_13.

Historia de usuario: HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte		
Tarea de ingeniería	**Caso de prueba**	**Tiempo**
TI_01 Implementar funcionalidad para verificar conexión a una red wifi, o por datos móviles	TC_01 Verificar si existe una conexión a internet activa	1s 357ms
	TC_02 Chequear que haya conexión activa de red wifi	2s 932ms
	TC_03 Chequear cuando no hay conexión activa de red wifi	683ms
	TC_04 Verificar que haya conexión activa de datos móviles	1s 584ms
	TC_05 Chequear cuando no hay conexión activa de datos móviles	506ms

	TC_06 Activar servicio para acceso a internet	659ms
	TC_07 Chequear cuando no existe servicio activo para acceso a internet	890ms
	TC_08 Verificar si existe conectividad y disponibilidad a datos móviles	1s 382ms
TI_02 Crear funcionalidad para consumo de servicios web mediante método post	TC_09 Verificar si fue exitoso el abrir una petición http	624ms
	TC_10 Verificar estado de la petición sea aceptada	3s 305ms
	TC_11 Chequear cuando la petición http fue rechazada	862ms
	TC_12 Verificar resultado de petición http respuesta no esté vacía	19s 343ms
	TC_13 Chequear cuando no hay resultado de consumo de servicio web	1s 987ms
TI_03 Crear funcionalidad para extracción y procesamiento de resultado devuelto por servicio web en hosting	TC_14 Verificar la disponibilidad de asientos de la respuesta traída de la petición http	3s 355ms
	TC_15 Verificar que el "Resultado" a la consulta de boletos disponibles sea una consulta correcta "CC"	8s 184ms
	TC_16 Verificar que el "Resultado" a la consulta de boletos disponibles sea una consulta incorrecta "CI"	6s 761ms
	TC_17 Verificar el número de asientos disponibles	609ms
	TC_18 Verificar la cantidad de asientos ocupados	610ms
Total	18 pruebas unitarias funcionales	55s 633ms

Realizado por: Wilmer B. 2018.

Las pruebas funcionales correspondientes al sprint 07, tienen un total de 33 pruebas unitarias automatizadas y se encuentran distribuidas en 2 historias de usuario, estas tienen un tiempo total de 55 segundos con 709 milisegundos en haber sido procesadas y proporcionar un resultado de aprobación o falla de estas.

Síntesis de pruebas unitarias automatizadas sprint 8

Tabla 28-3: Resultado de tiempo en pruebas automatizadas en la HU_14.

Historia de usuario: HU_14 Implementación de funcionalidad favoritos		
Tarea de ingeniería	**Caso de prueba**	**Tiempo**
TI_01 Implementar funcionalidad para extraer lista de favoritos	TC_01 Cargar lista de rutas favoritas extraída desde base de datos local	1s 456ms
	TC_02 Chuequear cuando no haya lista de favoritos extraída de la base de datos	1s 456ms
TI_02 Implementar funcionalidad parar extraer lista de objetos favoritos a lista de cadena de string	TC_03 Convertir lista de objeto de favoritos a lista de cadena string de favoritos	1s 181ms
TI_03 Implementar funcionalidad para agregar una ruta favorita a la lista	TC_04 Agregar ruta a favorito	1s 307ms
	TC_05 Verificar si dicho favorito ya existe	1s 259ms
	TC_07 Verificar si una ruta específica está registrada dentro de favorito	1s 281ms
TI_04 Implementar funcionalidad para extraer detalles de una ruta favorita en base a su id	TC_06 Obtener la ruta favorita a partir del id y la hora de la misma	1s 456ms
Total	7 pruebas unitarias funcionales	9s 274ms

Realizado por: Wilmer B. 2018.

Tabla 29-3: Resultado de tiempo en pruebas automatizadas en la HU_15.

Historia de usuario: HU_15 Implementación de funcionalidad llamada telefónica y cargar datos de agencia de cooperativa		
Tarea de ingeniería	**Caso de prueba**	**Tiempo**
TI_01 Implementar funcionalidad para extraer datos de una agencia de transporte	TC_01 Obtener datos de la agencia	805ms
	TC_02 Chequear cuando no existe datos de la agencia	503ms
TI_02 Implementar funcionalidad para realizar una llamada telefónica desde la aplicación	TC_03 Verificar permiso de llamada activo	603ms
	TC_04 Chequear cuando no existe permiso de llamada activo	1s 442ms
	TC_05 Convertir número string a formato de número para llamar	579ms
Total	5 pruebas unitarias funcionales	3s 932ms

Realizado por: Wilmer B. 2018.

Las pruebas funcionales correspondientes al sprint 08, tienen un total de 12 pruebas unitarias automatizadas y se encuentran distribuidas en 2 historias de usuario, estas tienen un tiempo total de 13 segundos con 206 milisegundos en haber sido procesadas y proporcionar un resultado de aprobación o falla de estas.

Síntesis de pruebas unitarias automatizadas sprint 9

Tabla 30-3: Resultado de tiempo en pruebas automatizadas en la HU_17.

Historia de usuario: HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting		
Tarea de ingeniería	**Caso de prueba**	**Tiempo**
TI_01 Implementar funcionalidad para verificar que	TC_01 Activar servicio de wifi	1s 383ms
	TC_02 Verificar conexión a la red activa	1s 582ms
	TC_03 Chequear cuando no hay conexión activa hacia la red	583ms

el dispositivo no esté conectado a datos móviles.	TC_04 Verificar que el servicio de conexión a la red esté disponible	1s 713ms
	TC_05 Verificar que el tipo de conexión sea wifi	2s 3ms
	TC_06 Verificar el tipo de conexión de red sea datos móviles	1s 284ms
TI_03 Implementar funcionalidad para extracción de resultados de petición de sincronización.	TC_07 Verificar que la respuesta del servicio web traiga un “Resultado” con “CC” de consulta correcta	1s 813ms
	TC_08 Verificar que la respuesta del servicio web traiga un “Resultado” con “CI” de consulta incorrecta	2s 190ms
TI_04 Implementar funcionalidad para la eliminación de contenido y tablas anteriores de la base de datos local	TC_09 Verificar que existan datos de la tabla “COOPERATIVA”	1s 408ms
	TC_10 Verificar que existan datos de la tabla “DEPENDENCIA”	1s 383ms
	TC_11 Verificar que existan datos de la tabla “TERMINAL”	1s 232ms
	TC_12 Verificar que existan datos de la tabla “RUTA”	1s 911ms
	TC_13 Verificar que existan datos de la tabla “HORARIO”	1s 708ms
	TC_14 Verificar que existan datos de la tabla “TERM_PERTENECE_DEPEN”	1s 532ms
	TC_15 Verificar que existan datos de la tabla “DEPEN_POSEE_RUTAS”	1s 487ms
	TC_16 Verificar que existan datos de la tabla “PROVINCIAS”	1s 810ms
	TC_17 Verificar que existan datos de la tabla “CIUDADES”	1s 207ms
	TC_18 Verificar que existan datos de la tabla “FAVORITO”	1s 333ms
	TC_19 Verificar eliminación de datos anteriores de la base de datos local	1s 760ms
TI_06 Implementar funcionalidad para inserción de datos de tablas locales con la	TC_20 Verificar inserción de datos de la tabla “COOPERATIVA” desde el servicio web en tablas locales de la base de datos	1s 158ms
	TC_21 Verificar inserción de datos de la tabla “DEPENDENCIA” desde el servicio web en tablas locales de la base de datos	1s 79ms
	TC_22 Verificar inserción de datos de la tabla “TERM_PERTENECE_DEPEN” desde el servicio web en tablas locales de la base de datos	1s 229ms

información extraída desde la respuesta del servicio web.	TC_23 Verificar inserción de datos de la tabla “TERMINAL” desde el servicio web en tablas locales de la base de datos	1s 223ms
	TC_24 Verificar inserción de datos de la tabla “DEPEN_POSEE_RUTAS” desde el servicio web en tablas locales de la base de datos	1s 234ms
	TC_25 Verificar inserción de datos de la tabla “RUTA” desde el servicio web en tablas locales de la base de datos	1s 934ms
	TC_26 Verificar inserción de datos de la tabla “HORARIO” desde el servicio web en tablas locales de la base de datos	1s 508ms
	TC_27 Verificar inserción de datos de la tabla “CIUDAD” desde el servicio web en tablas locales de la base de datos	1s 583ms
	TC_28 Verificar inserción de datos de la tabla “PROVINCIA” desde el servicio web en tablas locales de la base de datos	1s 409ms
Total	28 pruebas unitarias funcionales	41s 689ms

Realizado por: Wilmer B. 2018.

Las pruebas funcionales correspondientes al sprint 9, tienen un total de 7 pruebas unitarias automatizadas en una sola historia de usuario, las mismas que tienen un tiempo total de 41 segundos con 689 milisegundos en haber sido procesadas y proporcionar un resultado de aprobación o falla de estas.

Síntesis de pruebas unitarias automatizadas

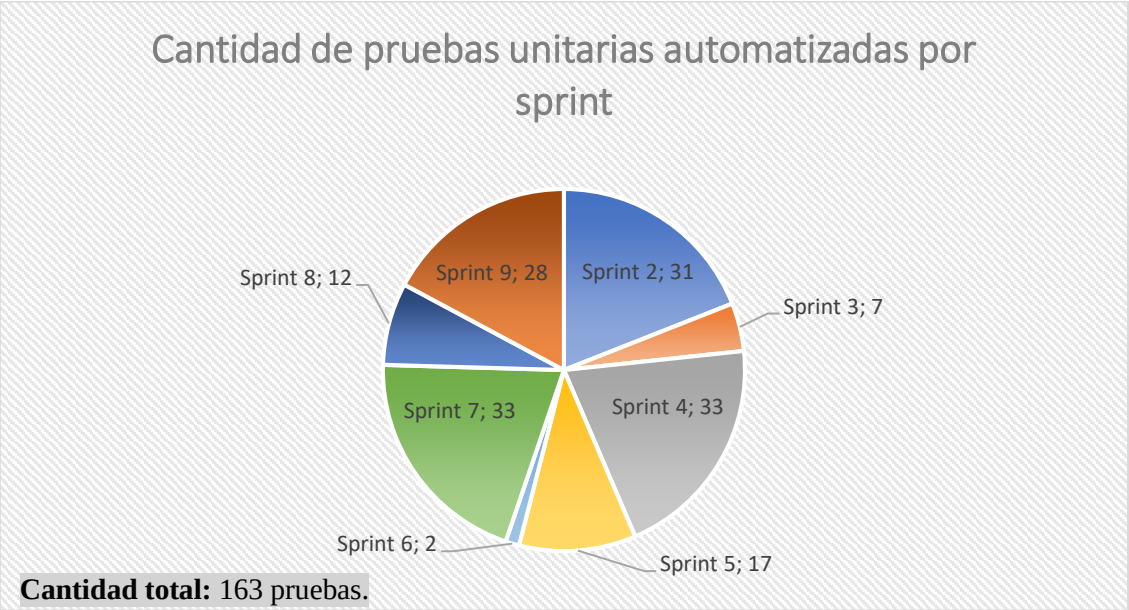


Gráfico 6-3: Cantidad de pruebas unitarias automatizadas por sprint.

Realizado por: Wilmer B. 2018.

Tabla 31-3: Cantidad de pruebas automatizadas.

Sprints	2	3	4	5	6	7	8	9	**7 sprints**
# Pruebas	31	7	33	17	2	33	12	28	**163**

Realizado por: Wilmer B. 2018.

En el **gráfico 6-3** se puede apreciar la distribución de cada uno de los sprints y el número o cantidad de pruebas automatizadas que han sido llevadas a cabo. Con un total de 163 pruebas unitarias automatizadas distribuidas a lo largo de 8 sprints que van del 2 al 9.

En la implementación de estas pruebas se ha obtenido los tiempos en cada uno de estas, así como también un resumen conciso del tiempo total, como el tiempo demorado en cada uno de los sprints, a continuación, en el **gráfico 7-3** podemos apreciar estos resultados.

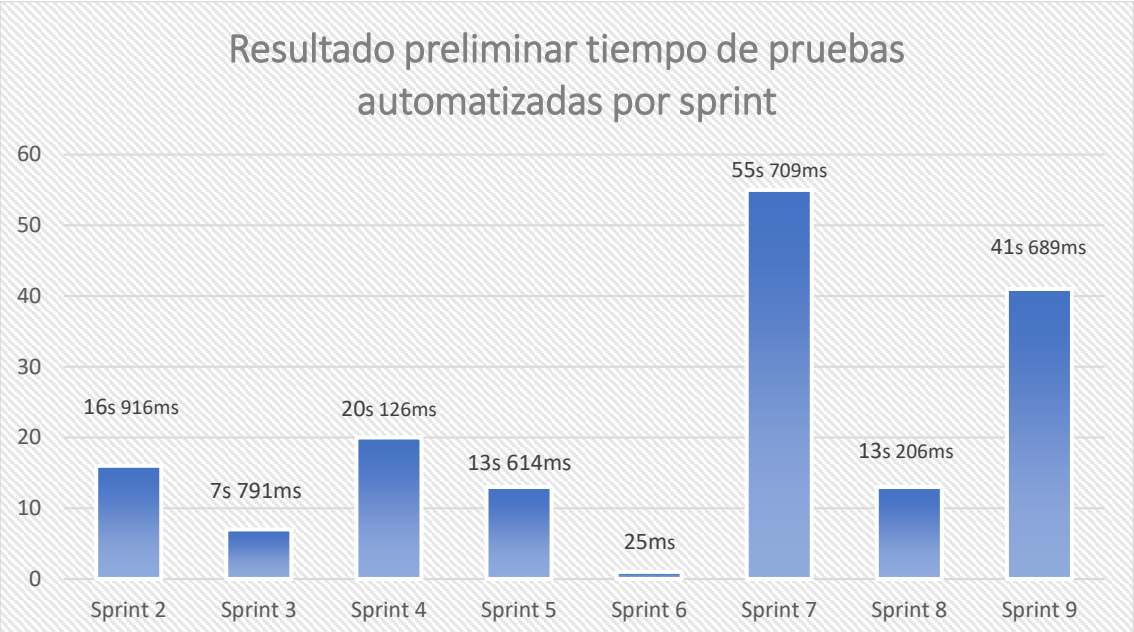


Gráfico 7-3: Tiempo preliminar de pruebas automatizadas por sprint.

Realizado por: Wilmer B. 2018.

Tabla 32-3: Tiempo preliminar de pruebas automatizadas por sprint.

Sprints	2	3	4	5	6	7	8	9	**7 sprints**
Tiempo	16s 916ms	7s 791ms	20s 126ms	13s 614ms	25ms	55s 709ms	13s 206ms	41s 689ms	**179s 76ms**

Realizado por: Wilmer B. 2018.

En el **gráfico 7-3** se puede observar cada uno de los tiempos tomados en la evaluación de pruebas automatizadas dentro de cada sprint, teniendo así un total de tiempo de demora igual a 169 segundos con 76 milisegundos.

Resultados finales en pruebas de aceptación

De igual manera para obtener el resultado final sobre las pruebas automatizadas se ha tenido en cuenta 3 variables que entran en juego al momento de evaluar las pruebas unitarias de forma automática, además que dos de estas ya fueron antes definidas en los resultados de pruebas de aceptación, a continuación, se detallas estas:

A. **Tiempo que se tarda en encender el emulador:** este tiempo ya antes ha sido definido en las pruebas de aceptación con 1 minuto con 52 segundos, pero para obtener la cantidad total de veces que se enciende el emulador se debe verificar todos los caminos de prueba evaluados desde la suite de pruebas Junit, esta información se encuentra respaldada en el

Anexo D a través de imágenes de pruebas junto a sus tiempos, cabe recalcar que cada una de las imágenes es un camino de evaluación o prueba.

B. **Tiempo que tarda el emulador en ejecutar la última versión del sistema:** este es uno de los tiempos ya antes definidos en las pruebas de aceptación con 13 segundos de demora. Y la cantidad de veces que se presentan estos tiempos son igual al número de caminos de evaluación manejados por la suite de Junit, esto debido a que no se ejecuta la última versión del sistema por cada prueba unitaria como es el caso de las pruebas de aceptación, sino que se ejecuta una vez por cada camino de evaluación de pruebas.

X. **Tiempo de ejecución de prueba unitaria:** Este tiempo se presenta en cada una de las pruebas unitarias cuando se manda a evaluar el sistema o una parte de este. Dicho tiempo se encuentra registrado en cada una de las tablas donde se documentaron estas, dentro de este documento estas tablas se encuentran en el **Anexo C** y su respaldo en imágenes con resultados capturados en el **Anexo D**.

La razón por la cual no se ha tomado en cuenta el tiempo de documentación de este tipo de pruebas es, porque esta documentación pasa a formar parte de la implementación o desarrollo de la prueba.

El total de caminos de evaluación manejados por la suite de Junit son 23. Información obtenida del **Anexo D**, donde cada imagen es un camino de evaluación y este conjunto de imágenes evalúa la totalidad de las pruebas unitarias las cuales son igual a 163.

Para obtener el valor total de la variable A, tenemos el dato que corresponde al tiempo en encenderse el emulador el cual corresponde a 1 minutos con 52 segundos.

Tiempo total de encendido del emulador: Tiempo de demora de encendido del emulador * Cantidad de veces que se enciende el emulador.

Variable A = 1 minuto con 52 segundos * 23 caminos de evaluación.

= 112 segundos * 23 caminos de evaluación

= 2576 segundos/ 42 minutos con 56 segundos

Ahora para obtener la cantidad de veces que se ejecuta la última versión del sistema tenemos el dato que corresponde al tiempo de demora de este proceso que es igual a 13 segundos.

Tiempo total de ejecución de la última versión del sistema = Cantidad de caminos de evaluación * Tiempo en ejecutarse la última versión.

Variable B = 13 segundos * 23 caminos de evaluación.

= 299 segundos/4 minutos con 59 segundos

Ahora se procederá a obtener el tiempo (segundos, milisegundos: ms) total de la ejecución de pruebas unitarias basadas en la información del **gráfico 8-3**, del cual tenemos el siguiente resumen:

Tabla 33-3: Resumen tiempos en pruebas automatizadas por sprint.

Sprint	2	3	4	5	6	7	8	9	**Total**
Tiempo (s/ms)	16s 916ms	7s 791ms	20s 126ms	13s 614ms	25ms	55s 709ms	13s 206ms	41s 689ms	**169s 76ms**

Realizado por: Wilmer B. 2018.

Ahora se obtendrá el tiempo total que se ha tomado en evaluar las pruebas de aceptación funcionales de manera automatizada para esto se obtendrá la suma de las 3 variables antes definidas:

Tiempo total empleado en pruebas de aceptación: A + B + X

Tiempo total = 2576 segundos + 299 segundos + 169 segundos con 76 milisegundos

= 3044segundos con 76 milisegundos/50min con 44segundos y 76 milisegundos.

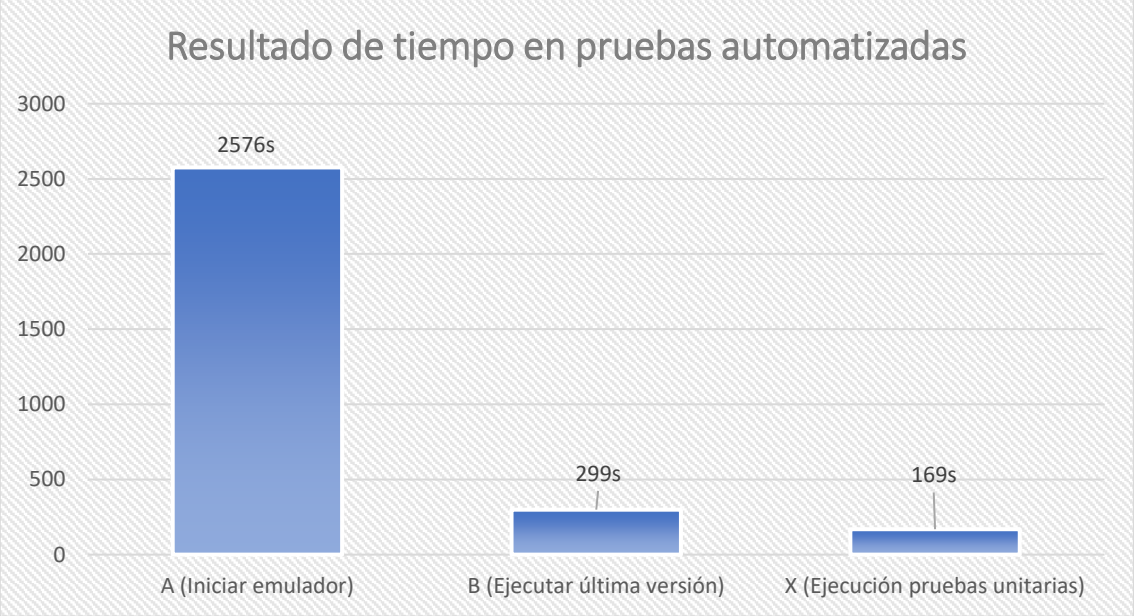


Gráfico 8-3: Resultado de tiempo en pruebas automatizadas.

Realizado por: Wilmer B. 2018.

Tabla 34-3: Variables de tiempo en pruebas automatizadas.

Variables	Tiempo
A (Iniciar emulador)	2576ms
B (Ejecutar última versión)	299ms
C (Evaluación pruebas unitarias)	169ms

Realizado por: Wilmer B. 2018.

En el **gráfico 8-3** se puede observar que el tiempo con mayor porcentaje pertenece a la iniciación del emulador con un 85% correspondiente a 2576 segundos del total del tiempo destino a la evaluación de todas las pruebas unitarias. Mientras que en la ejecución de la última versión tiene un 10% correspondiente a 299 segundos y por último se tiene la evaluación de pruebas unitarias que tan solo ocupa un 5% equivalente a 169 segundos con 76 milisegundos, este tiempo es muy corto debido a que la máquina es a encargada de llevar a cabo estas evaluaciones dando un resultado exitoso o fallido, pero de manera muy rápida a la que le tocaría analizar a una persona.

3.4 Comparación de pruebas manuales versus pruebas automáticas

Con los resultados obtenidos en el punto **3.2** y **3.3** se procederá a hacer un análisis en el cual se podrá notar las diferencias existentes en la aplicación de estos dos tipos de pruebas dentro de un proyecto software.

Cantidad de pruebas automatizadas y de aceptación

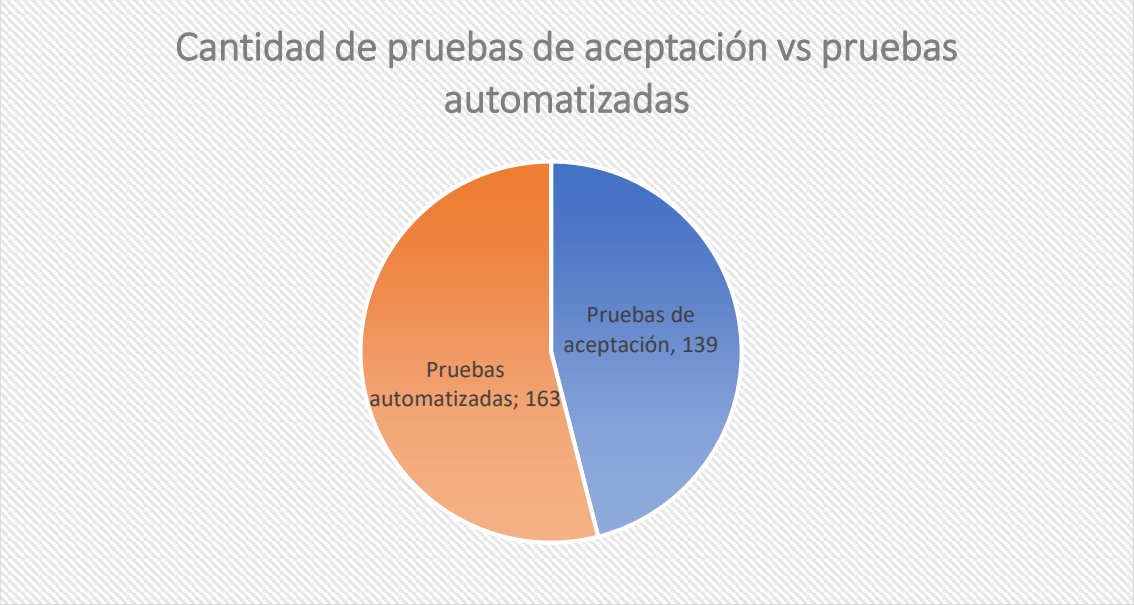


Gráfico 9-3: Cantidad de pruebas de aceptación vs pruebas automatizadas.

Realizado por: Wilmer B. 2018.

Tabla 35-3: Cantidad de pruebas manuales y automatizadas.

Pruebas	Automatizadas	Manuales
Cantidad	163	139

Realizado por: Wilmer B. 2018.

Como se puede observar en el **gráfico 9-3** se observa que la cantidad de pruebas automatizadas como las pruebas de aceptación no tienen igual cantidad. Existe una diferencia de 24 pruebas, ¿A qué se debe esto?, al momento de implementar las pruebas automatizadas, estas son desarrolladas de manera que evalué una sola funcionalidad en una prueba automatizada como por ejemplo en nuestro caso, se prueba la creación de base de datos tendríamos 1 prueba, la creación de cada una de las tablas de la base de datos serían en total 10 pruebas, la eliminación del contenido de cada una de las tablas de la base de datos serían 10 pruebas, etc. Mientras que al momento de realizar pruebas de aceptación se trata de probar un conjunto de pruebas en una sola, como por ejemplo el probar la creación de todas las tablas de la base de datos, lo que conllevaría a tener resumida 10 pruebas automatizadas en 1 sola prueba de aceptación. ¿Cuándo o por qué se hace esta reducción de evaluaciones en las pruebas de aceptación?, esto se hace con el fin de reducir el tiempo de evaluación de las funcionalidades que son similares o siguen un mismo proceso en la evaluación, ya que si evaluamos 1 prueba frente a otras 10 el tiempo que se reduce es significativo debido a que cada una de las pruebas conlleva un tiempo de ejecución de la última versión de la aplicación, además del tiempo que se sigue el paso a paso en la documentación de estas, dando como resultado el tener que aumentar tiempo a la hora de evaluar el sistema.

Tiempo empleado en pruebas automatizadas y de aceptación

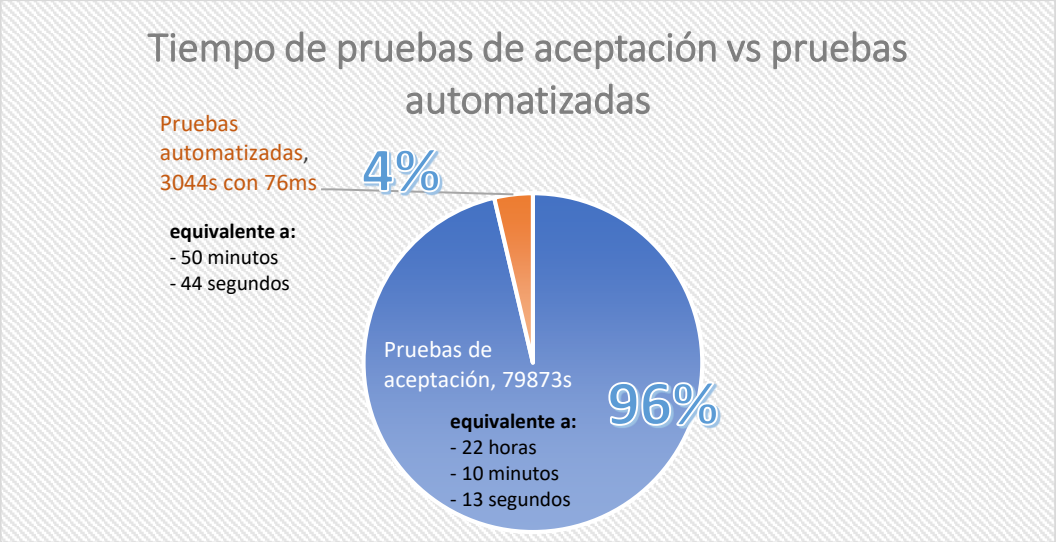


Gráfico 10-3: Tiempo de pruebas de aceptación vs pruebas automatizadas.

Realizado por: Wilmer B. 2018.

Tabla 36-3: Tiempo en pruebas manuales y automatizadas.

Pruebas	Automatizadas	Manuales
Tiempo	50min 44seg con 76ms	22h 10min con 13seg

Realizado por: Wilmer B. 2018.

A simple vista en el **gráfico 10-3** se puede observar una diferencia enorme que existe entre los tiempos de estos dos tipos de pruebas aplicadas en este proyecto, el cual es *76828s y 924ms*/ 1280min 28s y 924 ms/**21h 20min 28s y 924ms**, que es lo mismo a una diferencia de 92% entre estas dos pruebas. Este es un resultado bastante considerable, a la hora de evaluar las pruebas, pero una de las desventajas se presenta en el aumento de tiempo en la fase de desarrollo a la hora de implementar pruebas automatizadas, esto dentro de cada sprint en el que desarrollará dichas pruebas. Así como también en este caso para poder analizar dichos resultados entre estos tipos de pruebas se ha optado por documentar cada una de estas, cosa que en producción no se las haría ya que esto se maneja por medio de comentarios en el código de las pruebas automatizadas y el almacenamiento de captura de pantallas con los resultados de las pruebas, sus tiempos y el total de tiempo invertido, todo esto se puede hacer en el IDE Android Studio, con esto se reduciría aún más el tiempo que costó documentar cada una de estas pruebas dentro de este proyecto.

Uno de los puntos clave en los que se puede realmente diferenciar el tiempo entre estas pruebas son cuando el tester recolecta los tiempos que se demora en verificar visualmente y a criterio propio el resultado emitido por el sistema con el resultado esperado, esta observación hará que él tenga un criterio de aceptación o no, esto dentro de las pruebas de aceptación, mientras que en las pruebas automatizadas es el tiempo que el mismo sistema se demora en evaluar los caminos de prueba mostrando así la colección de resultados tanto si fueron exitosos y fallidos. Este es un factor realmente importante a la hora de obtener los resultados, ya que, en sí sería una confrontación entre la evaluación manual frente a la evaluación automatizada.

A continuación, se muestra una comparación de estos tiempos a la hora de evaluar el sistema tanto automatizando pruebas como aceptando pruebas manuales en base a criterio propio de una persona (tester). Para una mejor apreciación de estos tiempos serán mostrados en la unidad de tiempo igual a milisegundos (ms).

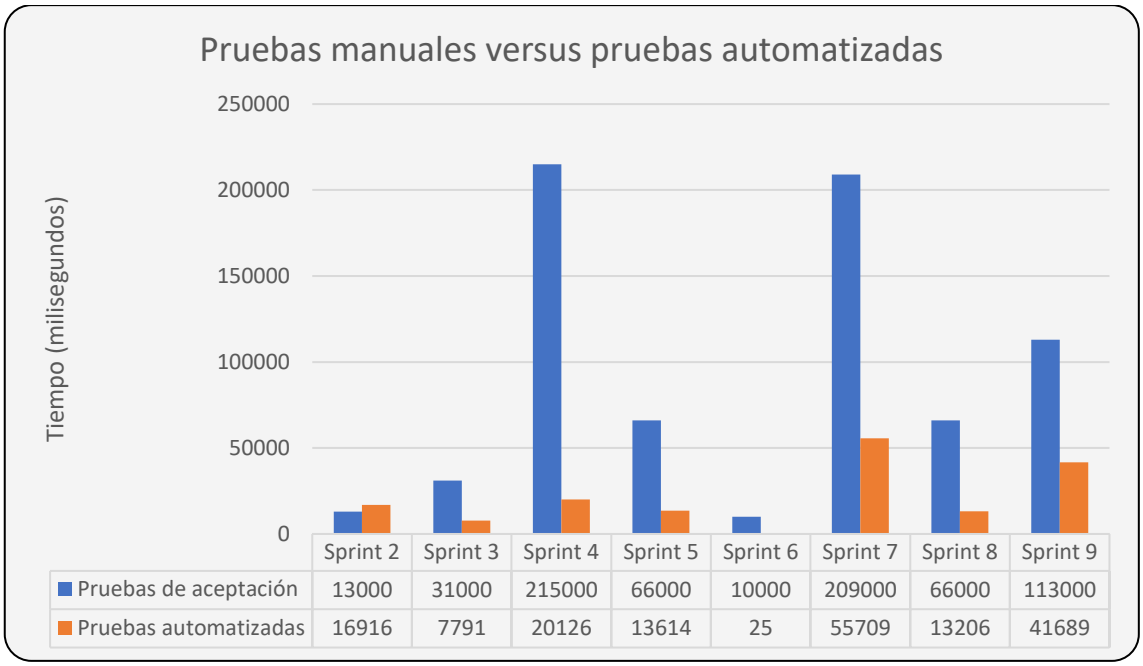


Gráfico 11-3: Pruebas manuales versus pruebas automatizadas.

Realizado por: Wilmer B. 2018.

Como se puede observar en las pruebas manuales se tiene un aumento del recurso tiempo mientras mayor sea la cantidad de información a visualizar, analizar, verificar y sacar un criterio sobre esta por parte del tester, además de tener inconvenientes que lo pueden confundir al manejar una gran cantidad de datos. Mientras que en las pruebas automatizadas este tiempo va a aumentar proporcionalmente a la cantidad de información que esté procesando.

CONCLUSIONES

- Se ha creado una suite de pruebas que gestiona caminos de evaluación, los cuales simulan en comportamiento de la aplicación con respecto a cada uno de los casos de prueba creados, pudiendo así evaluarse pruebas instrumentadas, de unidad local o estás dos en conjunto.

- Se ha determinado la cantidad de 139 pruebas funcionales correspondientes al 79% de total de pruebas empleadas en todo el proyecto. Las mismas que serán evaluadas tanto bajo pruebas manuales como automátizadas.

- Se ha implementado un total de 163 casos de prueba automatizados correspondientes explícitamente a pruebas funcionales, estas se encuentran distribuidas desde el segundo al noveno sprint planificado. Las mismas que alcanzaron un tiempo de ejecución equivalente al 4% del total del tiempo de pruebas en el proyecto.

- Se ha desarrollado la aplicación móvil siguiendo la metodología ágil scrum, de esta se ha obtenido un total de 10 sprints planificados en los cuales se distribuyeron un total de 18 historias de usuario y 13 historias técnicas. Además, se ha documentado un total de 69 tareas de ingeniería tanto de historias técnicas como de usuario y finalmente se ha documentado un total de 176 pruebas de aceptación aprobadas.

RECOMENDACIONES

- Profundizar en la aplicación de pruebas de integración y de interfaz dentro de plataformas móviles en conjunto con los frameworks destinados para este tipo de pruebas.

- Realizar un estudio minucioso sobre la geolocalización en equipos smartphone, esto con el fin de disminuir el consumo de recursos al momento de usar la localización gps o de red; al mismo tiempo tener un grado de precisión mayor al usar este servicio.

ÍNDICE DE ABREVIATURAS

2G	Segunda Generación
3G	Tercera Generación
4G	Cuarta Generación
API	Application Programming Interface
APK	Android Application Package
BD	Base de datos
CPU	Unidad central de proceso
CSS	Cascading Style Sheets
DOM	Modelo de objetos documentado
E/S	Entradas/salidas
GNU	GNU no es Unix
GPL	General Public License
GPS	Sistema de posicionamiento global
GUI	Interfaz gráfica de usuario
HTML	Lenguaje de marcas de hipertexto
HTTP	Protocolo de transferencia de hipertexto
IDE	Entorno de desarrollo integrado
IEC	Comisión Electrónica Internacional
IEEE	Instituto de Ingeniería Eléctrica y Electrónica
IOS	Sistema operativo Iphone
ISO	Organización Internacional de Normalización
JSON	JavaScript Object Notation
JVM	Máquina virtual de Java
MVP	Modelo Vista Presentador
NDK	Native Development Kit
ODBC	Open DataBase Connectivity
OEM	Original Equipment Manufacturer de Apple
OS	Sistema operativo
P2P	Conexión punto a punto
PHP	Hypertext Preprocessor
PNBV	Plan Nacional del Buen Vivir
QoS	Quality of service
SBP	Sistema bajo prueba
SDK	Software Development Kit

SQL	Lenguaje de consulta estructurada
TCP/IP	Protocolo de control de transmisión/Protocolo de internet
TDD	Test-Driven Development
UI	Interfaz de usuario
UML	Lenguaje unificado de modelado
URL	Localizador uniforme de recursos
Wi-Fi	Wireless Fidelity
XML	Lenguaje de marcado extensible

BIBLIOGRAFÍA

CADAVID, A.N., MARTÍNEZ, J.D.F. y VÉLEZ, J.M., 2013. Revisión de metodologías ágiles para el desarrollo de software. *Prospectiva*, vol. 11, no. 2, pp. 30–39. ISSN 1692-8261.

DELGADILLO, M.L.U., ABUD-FIGUEROA, M.A., PELÁEZ-CAMARENA, S.G., ALOR-HERNÁNDEZ, G. y GARCÍA, A.I.S., 2016. Propuesta de un modelo de integración de PSP y Scrum para mejorar la calidad del proceso de desarrollo en una MiPyME. *Research in Computing Science*, vol. 120, pp. 147–157. ISSN 1870-4069.

GAMMA, E. y BECK, K., 1999. JUnit: A cook's tour. *Java Report*, vol. 4, no. 5, pp. 27–38.

GAO, J., BAI, X., TSAI, W.-T. y UEHARA, T., 2014. Mobile application testing: a tutorial. *Computer*, vol. 47, no. 2, pp. 46–55. ISSN 0018-9162. DOI 10.1109/MC.2013.445.

GOOGLE LLC, 2017a. Cómo descargar Android Studio y SDK Tools | Android Studio. *Android Studio* [en línea]. [Consulta: 8 julio 2017]. Disponible en: https://developer.android.com/studio/index.html?hl=es-419.

GOOGLE LLC, 2017b. Conoce Android Studio | Android Studio. [en línea]. [Consulta: 8 julio 2017]. Disponible en: https://developer.android.com/studio/intro/index.html?hl=es-419.

GOOGLE LLC, 2017c. Guía del desarrollador | Google Maps Geocoding API. *Google Developers* [en línea]. [Consulta: 29 marzo 2018]. Disponible en: https://developers.google.com/maps/documentation/geocoding/intro?hl=es.

GOOGLE LLC, 2018a. Fundamentals of Testing | Android Developers. *Fundamentals of testing* [en línea]. [Consulta: 28 marzo 2018]. Disponible en: https://developer.android.com/training/testing/fundamentals.html?hl=es-419#medium-tests.

GOOGLE LLC, 2018b. Primeros pasos | Google Maps Geocoding API. *Google Developers* [en línea]. [Consulta: 28 marzo 2018]. Disponible en: https://developers.google.com/maps/documentation/geocoding/start?hl=es.

GOOGLE LLC, 2018c. Probar tu app | Android Studio. *Probar tu APP* [en línea]. [Consulta: 26 marzo 2018]. Disponible en: https://developer.android.com/studio/test/index.html?hl=es-419.

HAUGHEE, E., 2013. *Instant Sublime Text starter: learn to efficiently author software, blog posts, or any other text with Sublime Text 2* [en línea]. 1. Birmingham: Packt Pub. [Consulta: 21 marzo 2018]. ISBN 978-1-84969-393-6. Disponible en: http://site.ebrary.com/id/10654580.

HTTP://WWW.WAMPSERVER.COM, 2018. WampServer. *WampServer* [en línea]. [Consulta: 21 marzo 2018]. Disponible en: http://www.wampserver.com/en/.

IEEE COMPUTER SOCIETY, SOFTWARE & SYSTEMS ENGINEERING STANDARDS COMMITTEE, INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS y IEEE-SA STANDARDS BOARD, 2008. *IEEE standard for software and system test documentation* [en línea]. New York, NY: Institute for Electrical and Electronics Engineers. [Consulta: 3 enero 2018]. ISBN 978-0-7381-5746-7. Disponible en: http://ieeexplore.ieee.org/servlet/opac?punumber=4578271.

ISO, I. 1 y SC 7, 2013. ISO / IEC / IEEE 29119-1: 2013 - Ingeniería de software y sistemas - Pruebas de software - Parte 1: Conceptos y definiciones. *ISO/IEC/IEEE 29119-1:2013* [en línea]. [Consulta: 3 enero 2018]. Disponible en: https://www.iso.org/standard/45142.html.

KARLESKY, M.J., BEREZA, W.I. y ERICKSON, C.B., 2006. Effective test driven development for embedded software. *Electro/information Technology, 2006 IEEE International Conference on* [en línea]. East Lansing, MI, USA: IEEE, pp. 382–387. ISBN 0-7803-9592-1. DOI 10.1109/EIT.2006.252188. Disponible en: http://ieeexplore.ieee.org/document/4017725/.

KNIBERG, H., COHN, M. y SUTHERLAND, J., 2015. *Scrum and XP from the Trenches: how we do Scrum*. S.l.: C4Media. ISBN 978-1-4303-2264-1.

LEWIS, W.E. y VEERAPILLAI, G., 2005. *Software testing and continuous quality improvement* [en línea]. 2nd ed. Boca Raton: Auerbach Publications. ISBN 978-0-8493-2524-3. Disponible en: https://books.google.com.ec/books/about/Software_Testing_and_Continuous_Quality.html?id=ZBX6_8wqGikC&redir_esc=y. QA76.76.T48 L495 2005

LIN, F. y YE, W., 2009. Operating System Battle in the Ecosystem of Smartphone Industry. *Information Engineering and Electronic Commerce, 2009. IEEC '09. International Symposium on* [en línea]. Ternopil, Ukraine: IEEE, pp. 617-621. [Consulta: 20 mayo 2018]. ISBN 978-0-7695-3686-6. DOI 10.1109/IEEC.2009.136. Disponible en: http://ieeexplore.ieee.org/document/5175193/.

MALAVE POLANCO, K. y BEAUPERTHUY TAIBO, J.L., 2011. Android. *Negotium*, vol. 7, no. 19, pp. 79-96. ISSN 1856-1810.

MARIAN JURECZKO, I.K., MARCIN KASPRZYK, P.C. y JAROSŁAW BERŁOWSKI, 2016. Highly Automated Agile Testing Process: An Industrial Case Study | Directory of Open Access Journals. *Wroclaw University of Science and Technology*, vol. 1, pp. 69-87. ISSN 1897-7979.

MEI, H., HAO, D., ZHANG, Lingming, ZHANG, Lu, ZHOU, J. y ROTHERMEL, G., 2012. A Static Approach to Prioritizing JUnit Test Cases. *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1258-1275. ISSN 0098-5589. DOI 10.1109/TSE.2011.106.

ORACLE CORPORATION, 2018a. MySQL| Oracle. *World's Most Popular Open Source Database* [en línea]. [Consulta: 20 marzo 2018]. Disponible en: https://www.oracle.com/mysql/index.html.

ORACLE CORPORATION, 2018b. MySQL :: MySQL 5.7 Reference Manual. *MySql* [en línea]. [Consulta: 20 marzo 2018]. Disponible en: https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html.

ORACLE CORPORATION, 2018c. MySQL :: MySQL 5.7 Reference Manual. *MySql* [en línea]. [Consulta: 12 junio 2018]. Disponible en: https://dev.mysql.com/doc/refman/5.7/en/features.html.

PARANJ, B., 2017. *Test Driven Development in Ruby* [en línea]. 2. Atlanta, Georgia, USA: Apress. 1, 1. ISBN 978-1-4842-2638-4. Disponible en: https://books.google.com.ec/books?id=mYtcDgAAQBAJ&printsec=frontcover&dq=test-driven+development&hl=es&sa=X&ved=0ahUKEwiYwvmmqYjaAhXEzlkKHZhzBHcQ6AEIJTAA#v=onepage&q&f=false.

PAU, V.C., MIHAILESCU, M.I. y STANESCU, O., 2010. Model View Presenter Design Pattern. *Journal of Computer Science and Control Systems*, vol. 3, no. 1, pp. 173.

RUBIN, K.S., 2012. *Essential Scrum: a practical guide to the most popular agile process* [en línea]. 1ST. Upper Saddle River, NJ: Addison-Wesley. 1, 1. ISBN 978-0-13-704329-3. Disponible en: https://dl.acm.org/citation.cfm?id=2380978. QA76.76.D47 R824 2012

SOOD, G., 2016. *Scale Test-Driven Development* [en línea]. 1. Birmingham - Mumbai: Packt Pub. 1, 1. ISBN 978-1-78646-467-5. Disponible en: https://books.google.com.ec/books?id=cZjcDgAAQBAJ&pg=PA141&lpg=PA141&dq=Scala+Test-Driven+Development&source=bl&ots=8iNqmRdxOi&sig=Sjivvfn_WcDkkKqjfQZgj5ltkXM&hl=es&sa=X&ved=0ahUKEwiz_KuVx4jaAhUS0lMKHa13B2c4FBDoAQhlMAg#v=onepage&q=Scala%20Test-Driven%20Development&f=false.

SPILLNER, A., LINZ, T. y SCHAEFER, H., 2014. *Software testing foundations: a study guide for the certified tester exam: foundation level, ISTQB compliant* [en línea]. 4th edition. Santa Barbara, CA: Rocky Nook, Inc. ISBN 978-1-937538-42-2. Disponible en: https://www.kobo.com/us/es/ebook/software-testing-foundations-4th-edition. QA76.76.T48 S66 2014

SQLITE.ORG, 2017. About SQLite. [en línea]. [Consulta: 8 julio 2017]. Disponible en: http://www.sqlite.org/about.html.

SQLITE.ORG, 2017. Features Of SQLite. *Features Of SQLite* [en línea]. [Consulta: 19 junio 2017]. Disponible en: https://www.sqlite.org/features.html.

TRIGAS GALLEGO, M., 2012. *Metodología Scrum* [en línea]. 2012. S.l.: s.n. Disponible en: http://www.academia.edu/download/39164786/mtrigasTFC0612memoria_1.pdf.

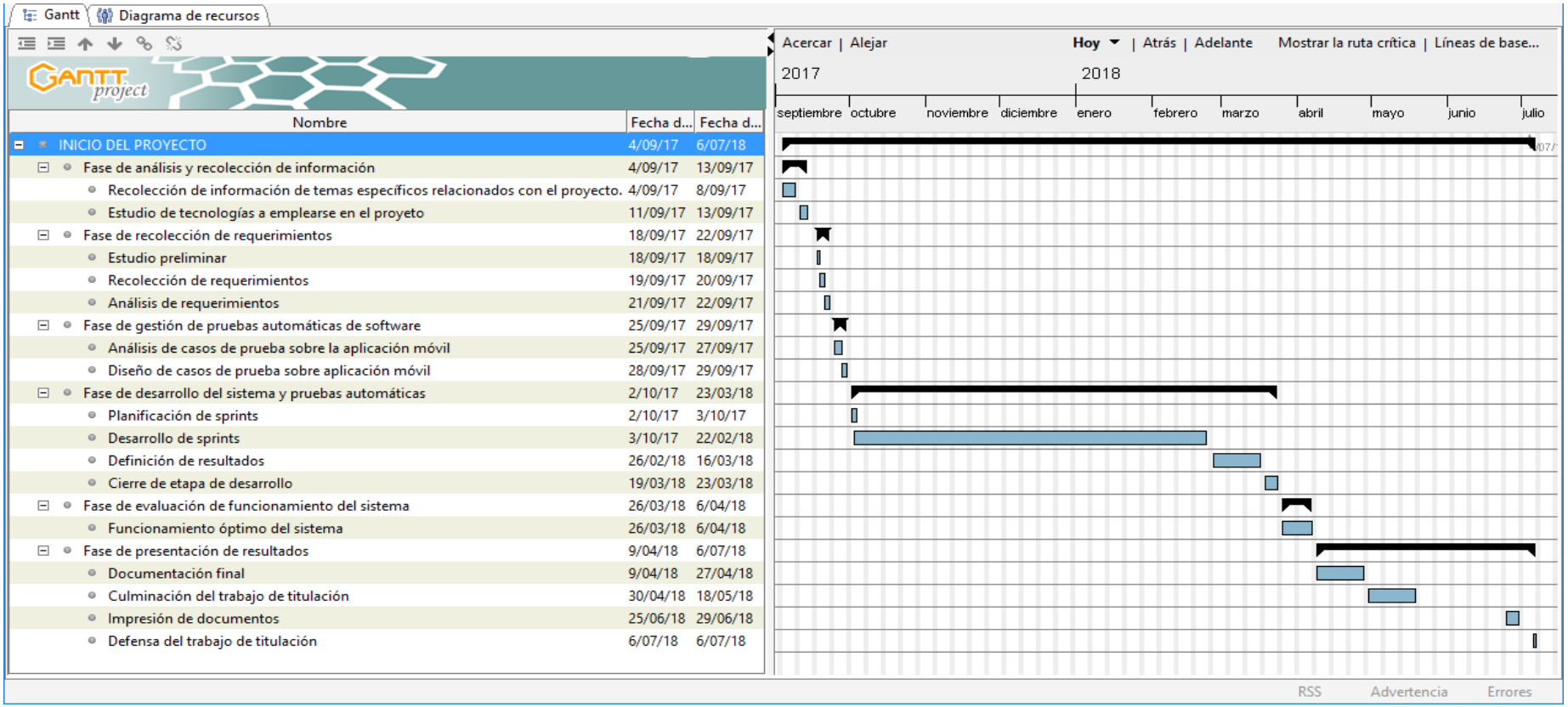
WICK, M., STEVENSON, D. y WAGNER, P., 2005. Using testing and JUnit across the curriculum. *ACM SIGCSE Bulletin* [en línea]. St. Louis, Missouri, USA: ACM, pp. 236–240. ISBN 1-58113-997-7. DOI 10.1145/1047124.1047427. Disponible en: https://dl.acm.org/citation.cfm?id=1047427.

YAGÜE, A. y GARBAJOSA, J., 2009. Las pruebas en metodologías ágiles y convencionales: papeles diferentes. *Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos*, vol. 3, no. 4, pp. 73. ISSN 1988-3455.

ZHANG, Y. y LUO, Y., 2010. An architecture and implement model for Model-View-Presenter pattern. *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on* [en línea]. Chengdu, China: IEEE, pp. 532–536. ISBN 978-1-4244-5540-9. DOI 10.1109/ICCSIT.2010.5565090. Disponible en: http://ieeexplore.ieee.org/abstract/document/5565090/.

ANEXOS

Anexo A: Planificación de actividades



Anexo A: Plan de actividades.

Realizado por: Wilmer B. 2018.

Anexo B: Historias de usuario/técnicas

Sprint 2

Tabla historia de usuario 01.

HISTORIA DE USUARIO		
Id: HU_01	**Nombre:** Codificación de scripts SQL para transacciones en base de datos	
Descripción: Como programador necesito crear cada uno de los scripts necesarios para realizar transacciones sql a la base de datos local, los mismos que se tendrán en archivos de respaldo para el cliente.		
Usuario: Wilmer Barrera	**Sprint:** 2	
Fecha inicio: 16/10/17	**Fecha fin:** 25/10/17	**Esfuerzo:** 40

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Implementación de script para creación de tablas
TI_02	Implementación de script para eliminación de contenido de tablas
TI_03	Implementación de script para eliminación de tablas
TI_04	Implementación de funcionalidad para crear base de datos y conexión a esta.

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 01.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 2
Nombre de historia usuario: HU_01 Codificación de scripts SQL para transacciones en base de datos		
Nombre tarea: Implementación de script para creación de tablas.		
Fecha inicio: 16/10/17		**Fecha fin:**17/10/17
Descripción: Como desarrollador necesito crear los scripts necesarios para la creación de cada una de las tablas de la base de datos		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_07	Verificar creación de tabla: COOPERATIVA, DEPENDENCIA, DEPEN_POSEE_RUTAS, HORARIO, RUTA, TERMINAL,

	TERM_PERTENECE_DEP, PROVINCIAS, CIUDADES, FAVORITO en base de datos local
PA_08	Mostrar advertencia cuando una tabla no se ha creado.

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 07.

PRUEBA DE ACEPTACIÓN	
Id: PA_07	**Nombre:** Verificar creación de tabla: COOPERATIVA, DEPENDENCIA, DEPEN_POSEE_RUTAS, HORARIO, RUTA, TERMINAL, TERM_PERTENECE_DEP, PROVINCIAS, CIUDADES, FAVORITO en base de datos local
Tarea de ingeniería: TI_01 Implementación de script para creación de tablas	
Descripción: Para poder tener un respaldo de información de os transportes y sus rutas, como una manera de trabajar offline se deben crear las tablas necesarias en las que se almacene dicha información.	
Responsable: Wilmer Barrera	**Fecha:** 17/10/17
Condición de ejecución: - Tener instalado el emulador - Tener instalada la última versión de la app en el emulador - Tener creado los archivos de conexión y scripts de la base de datos	
Pasos de ejecución: - Ejecutar el emulador - Ejecutar la última versión de la app - Ejecutar la parte de la aplicación en la que se ejecuta los scripts de creación de las tablas: COOPERATIVA, DEPENDENCIA, DEPEN_POSEE_RUTAS, HORARIO, RUTA, TERMINAL, TERM_PERTENECE_DEP, PROVINCIAS, CIUDADES, FAVORITO.	
Resultado esperado: cuando el proceso de creación de las tablas de la base de datos local sea correcto la aplicación seguirá ejecutándose con normalidad, mientras se muestra que este proceso se llevó a cabo con éxito a través de la consola	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 08.

PRUEBA DE ACEPTACIÓN	
Id: PA_08	**Nombre:** Mostrar advertencia cuando una tabla no se ha creado.
Tarea de ingeniería: TI_01 Implementación de script para creación de tablas	
Descripción: Cuando no se puede llevar a cabo la creación de una u otra tabla, es necesario mostrar un mensaje dando a conocer que este proceso fue fallido.	
Responsable: Wilmer Barrera	**Fecha:** 17/10/17
Condición de ejecución: - Tener instalado un emulador - Tener instalada la última versión de la app en el emulador - Tener creado los archivos de conexión a la base de datos, así como los scripts para creación de tablas.	
Pasos de ejecución: - Alterar el script de creación de tablas de manera que falle a la hora de ejecutar la creación de una tabla - Ejecutar el emulador - Ejecutar la parte de la app donde se crean las tablas.	
Resultado esperado: Al no poder crearse la primera tabla el sistema lanzará un mensaje donde muestra una advertencia diciendo que no se pudo crear tal tabla.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 02 de historia de usuario 01.

TAREA DE INGENIERÍA		
Id: TI_02	**Tipo de tarea:** desarrollo	**Sprint:** 2
Nombre de historia usuario: HU_01 Codificación de scripts SQL para transacciones en base de datos		
Nombre tarea: Implementación de script para eliminación de contenido de tablas.		
Fecha inicio: 18/10/17		**Fecha fin:**18/10/17
Descripción: Como desarrollador necesito crear los scripts necesarios para la eliminación del contenido de cada una de las tablas de la base de datos		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_09	Verificar que una tabla se encuentre creada	

PA_10	Verificar que se haya eliminado el contenido de la tabla: COOPERATIVA, DEPENDENCIA, DEPEN_POSEE_RUTAS, HORARIO, RUTA, TERMINAL, TERM_PERTENECE_DEP, PROVINCIAS, CIUDADES, FAVORITO de la base de datos local
PA_11	Mostrar advertencia cuando eliminación de contenido fue fallido en una tabla

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 09.

PRUEBA DE ACEPTACIÓN	
Id: PA_09	**Nombre:** Verificar que una tabla se encuentre creada
Tarea de ingeniería: TI_02 Implementación de script para eliminación de contenido de tablas	
Descripción: Para poder eliminar una tabla o su contenido, antes debemos verificar que esta tabla exista para poder tomar cualquier acción sobre ella.	
Responsable: Wilmer Barrera	**Fecha:** 18/10/17
Condición de ejecución: - Tener instalado un emulador - Tener instalada la última versión de la app - Tener creado las tablas necesarias en la base de datos - Tener creado los archivos de conexión a la base de datos	
Pasos de ejecución: - Alterar el script de verificación de existencia de una tabla para que verifique si existe la tabla COOPERATIVA - Ejecutar el emulador - Ejecutar la parte de la app donde se ejecute el script de verificación de existencia de la tabla COOPERATIVA	
Resultado esperado: Al ejecutarse este proceso de verificación de existencia de la tabla COOPERATIVA, si en verdad existe dicha tabla la app seguirá ejecutándose de manera normal ya que este proceso da como resultado una respuesta verdadera, mientras que si no se encuentra dicha tabla mostrará un mensaje advirtiendo que no existe la tabla.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 10.

PRUEBA DE ACEPTACIÓN	
Id: PA_10	**Nombre:** Verificar que se haya eliminado el contenido de la tabla: COOPERATIVA, DEPENDENCIA, DEPEN_POSEE_RUTAS, HORARIO, RUTA, TERMINAL, TERM_PERTENECE_DEP, PROVINCIAS, CIUDADES, FAVORITO de la base de datos local
Tarea de ingeniería: TI_02 Implementación de script para eliminación de contenido de tablas	
Descripción: Para poder sincronizar las bases de datos de la aplicación y la del servicio de hosting. La base de datos local deberá eliminar su contenido anterior antes de volver a subir nuevo contenido.	
Responsable: Wilmer Barrera	**Fecha:** 18/10/17
Condición de ejecución: - Tener instalado el emulador - Tener instalada la última versión de la app - Tener creados los archivos de conexión a la base de datos y scripts	
Pasos de ejecución: - Ejecutar el emulador - Ejecutar la parte de la app donde se ejecuten los scripts para eliminar el contenido de las tablas: COOPERATIVA, DEPENDENCIA, DEPEN_POSEE_RUTAS, HORARIO, RUTA, TERMINAL, TERM_PERTENECE_DEP, PROVINCIAS, CIUDADES, FAVORITO de la base de datos local.	
Resultado esperado: Al eliminar el contenido de cualquier tabla, este proceso devolverá un valor verdadero o falso, en caso de que cumpla o no con la eliminación de contenido de la tabla. En caso de que la ejecución del proceso para eliminar el contenido de las tablas se lleve a cabo sin problemas la app seguirá ejecutándose normalmente.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 11.

PRUEBA DE ACEPTACIÓN	
Id: PA_11	**Nombre:** Mostrar advertencia cuando eliminación de contenido fue fallido en una tabla
Tarea de ingeniería: TI_02 Implementación de script para eliminación de contenido de tablas	
Descripción: Cuando se necesite sincronizar las bases de datos se necesitará eliminar el contenido de las tablas de la base de datos. Pero cuando no se ha podido eliminar el contenido de una tabla el sistema deberá reaccionar ante esto.	

Responsable: Wilmer Barrera	**Fecha:** 18/10/17
Condición de ejecución: - Tener instalado un emulador - Tener creada la base de datos - Tener los archivos de conexiones creados - Tener instalada la última versión de la app	
Pasos de ejecución: - Alterar el código del script de eliminación de contenido de la tabla COOPERATIVA - Ejecutar la app en el emulador - Ejecutar la parte de la app donde se ejecute el script para eliminar contenido de la tabla COOPERATIVA	
Resultado esperado: cuando no se puede eliminar el contenido de la tabla COOPERATIVA la aplicación mostrará una advertencia en la que dirá que el proceso de eliminación de contenido ja sido fallido.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 03 de historia de usuario 01.

TAREA DE INGENIERÍA		
Id: TI_03	**Tipo de tarea:** desarrollo	**Sprint:** 2
Nombre de historia usuario: HU_01 Codificación de scripts SQL para transacciones en base de datos		
Nombre tarea: Implementación de script para eliminación de tablas.		
Fecha inicio: 19/10/17		**Fecha fin:**20/10/17
Descripción: Como desarrollador necesito crear los scripts necesarios para la eliminación definitiva de cada una de las tablas de la base de datos		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_12	Verificar que la tabla: COOPERATIVA, DEPENDENCIA, DEPEN_POSEE_RUTAS, HORARIO, RUTA, TERMINAL, TERM_PERTENECE_DEP, PROVINCIAS, CIUDADES, FAVORITO se haya eliminado correctamente de la base de datos local
PA_13	Mostrar mensaje cuando una tabla no se haya podido eliminar.

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 12.

PRUEBA DE ACEPTACIÓN	
Id: PA_12	**Nombre:** Verificar que la tabla: COOPERATIVA, DEPENDENCIA, DEPEN_POSEE_RUTAS, HORARIO, RUTA, TERMINAL, TERM_PERTENECE_DEP, PROVINCIAS, CIUDADES, FAVORITO se haya eliminado correctamente de la base de datos local
Tarea de ingeniería: TI_03 Implementación de script para eliminación de tablas	
Descripción: Para poder almacenar cada uno de los registros de las tablas en el servicio de hosting es necesario crear una base de datos con los campos más necesario al igual que las tablas para poder almacenar la información que le interesa al usuario.	
Responsable: Wilmer Barrera	**Fecha:** 20/10/17
Condición de ejecución: - Tener instalado el emulador - Tener creado los archivos de creación y conexión a la base de datos	
Pasos de ejecución: - Ejecutar el emulador - Ejecutar la última versión de la app en el emulador - Ejecutar parte de la aplicación que contiene los scripts de creación de las tablas: COOPERATIVA, DEPENDENCIA, DEPEN_POSEE_RUTAS, HORARIO, RUTA, TERMINAL, TERM_PERTENECE_DEP, PROVINCIAS, CIUDADES, FAVORITO.	
Resultado esperado: cuando se elimine cada una de las tablas este proceso devolverá un valor de verdadero o falso en caso de que se eliminó o no alguna tabla.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 13.

PRUEBA DE ACEPTACIÓN	
Id: PA_13	**Nombre:** Mostrar mensaje cuando una tabla no se haya podido eliminar.
Tarea de ingeniería: TI_03 Implementación de script para eliminación de tablas	
Descripción: Cuando se necesita eliminar una tabla de la base de datos, y este proceso no se lleva a cabo con éxito.	
Responsable: Wilmer Barrera	**Fecha:** 20/10/17
Condición de ejecución: - Tener instalado un emulador - Tener creado los archivos de conexión a la base de datos	

- Tener instalado la última versión de la app
Pasos de ejecución: - Alterar el script de eliminación de la tabla COOPERATIVA de manera que falle este cuando se quiera eliminar dicha tabla. - Mandar a ejecutar el emulador - Mandar a ejecutar la parte de la app donde se ejecute los scripts de eliminación de la tabla COOPERATIVA
Resultado esperado: Cuando no se pueda eliminar la tabla COOPERATIVA el sistema mostrará un mensaje de advertencia donde dice que el proceso de eliminación de la tabla ha sido fallido.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 04 de historia de usuario 01.

TAREA DE INGENIERÍA		
Id: TI_04	**Tipo de tarea:** desarrollo	**Sprint:** 2
Nombre de historia usuario: HU_01 Codificación de scripts SQL para transacciones en base de datos		
Nombre tarea: Implementación de funcionalidad para crear base de datos y conexión a esta.		
Fecha inicio: 23/10/17		**Fecha fin:**25/10/17
Descripción: Como desarrollador necesito implementar la función que permita crear base de datos local en la aplicación, así como permitir tener una conexión a esta		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_14	Verificar que la base de datos se creó correctamente
PA_15	Mostrar mensaje cuando la base de datos no se pudo crear correctamente
PA_16	Verificar que se ejecute correctamente la conexión a la base de datos
PA_17	Mostrar mensaje cuando no se puede conectar a la base de datos

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 14.

PRUEBA DE ACEPTACIÓN	
Id: PA_14	**Nombre:** Verificar que la base de datos se creó correctamente
Tarea de ingeniería: TI_04 Implementación de funcionalidad para crear base de datos y conexión a esta	
Descripción: Para poder almacenar la información de los transportes es necesario implementar una base de datos local, en la cual sirva para poder extraer estos con la aplicación y a la vez funcione de manera offline.	
Responsable: Wilmer Barrera	**Fecha:** 25/10/17
Condición de ejecución: - Tener instalado el emulador - Tener creado los archivos de conexión y creación de la base de datos local - Tener instalada la última versión de la app.	
Pasos de ejecución: - Ejecutar el emulador - Cargar la última versión del sistema - Verificar que no muestre o presente en pantalla ningún mensaje cuando no se crea la base de datos.	
Resultado esperado: Cuando se cree la base de datos dentro de la app, dentro de la app se crea un objeto de la base de datos, la cual será null en caso no haberse creado y una dirección de memoria en caso contrario.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 15.

PRUEBA DE ACEPTACIÓN	
Id: PA_15	**Nombre:** Mostrar mensaje cuando la base de datos no se pudo crear correctamente
Tarea de ingeniería: TI_04 Implementación de funcionalidad para crear base de datos y conexión a esta	
Descripción: Cuando no se tenga éxito en la creación de base de datos, la aplicación deberá responder ante esta situación.	
Responsable: Wilmer Barrera	**Fecha:** 25/10/17
Condición de ejecución: - Tener instalado un emulador - Tener creado los archivos de conexión de la base de datos	

Pasos de ejecución: - Ejecutar el emulador - Mandar a ejecutar la parte de la aplicación donde está el script de creación de la base de datos.
Resultado esperado: cuando no se cree la base de datos por algún motivo, la aplicación mostrará un mensaje advirtiendo sobre lo sucedido.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 16.

PRUEBA DE ACEPTACIÓN	
Id: PA_16	**Nombre:** Verificar que se ejecute correctamente la conexión a la base de datos
Tarea de ingeniería: TI_04 Implementación de funcionalidad para crear base de datos y conexión a esta	
Descripción: para poder acceder a la información de la base de datos local se necesita crear un archivo en el cual se realice la conexión con la misma.	
Responsable: Wilmer Barrera	**Fecha:** 25/10/17
Condición de ejecución: - Tener instalado un emulador - Tener creado la clase de conexión a la base de datos local	
Pasos de ejecución: - Ejecutar el emulador - Ejecutar la última versión de la app en el emulador - Ejecutar la app de manera que se ejecute la clase de conexión a la base de datos	
Resultado esperado: cuando necesitemos acceder a la base de datos crearemos objetos de la clase base de datos en el cual si la conexión con la misma fue exitosa se creará el objeto, caso contrario no. Caso que sea exitoso la app seguirá ejecutándose normalmente.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 17.

PRUEBA DE ACEPTACIÓN	
Id: PA_17	**Nombre:** Mostrar mensaje cuando no se puede conectar a la base de datos
Tarea de ingeniería: TI_04 Implementación de funcionalidad para crear base de datos y conexión a esta	
Descripción: Cuando exista problemas de conexión a la base de datos la aplicación debe reaccionar antes esta situación.	
Responsable: Wilmer Barrera	**Fecha:** 25/10/17
Condición de ejecución: - Tener instalado un emulador - Tener creado la clase de conexión a la base de datos local	
Pasos de ejecución: - Alterar el script de conexión de la conexión de la base de datos - Ejecutar el emulador - Ejecutar la app para que se ejecute la conexión a la base de datos	
Resultado esperado: al no haber una conexión exitosa con la base de datos, el sistema mostrará un mensaje notificando que no se puede conectar a la base de datos local.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Sprint 3

Tabla historia de usuario 02.

HISTORIA DE USUARIO		
Id: HU_02	**Nombre:** Implementación de interfaz para consulta de horarios por cooperativa de transporte	
Descripción: Como desarrollador necesito implementar la interfaz de acuerdo a lo requerido por el cliente, con el fin de mostrar la información de horarios de una ruta específica de una cooperativa de transporte.		
Usuario: Wilmer Barrera	**Sprint:** 3	
Fecha inicio: 31/10/17	**Fecha fin:** 01/11/17	**Esfuerzo:** 10

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Diseño de interfaz para consulta de horarios por cooperativa de transporte

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 02.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 3
Nombre de historia usuario: HU_02 Implementación de interfaz para consulta de horarios por cooperativa de transporte		
Nombre tarea: Diseño de interfaz para consulta de horarios por cooperativa de transporte.		
Fecha inicio: 31/10/17		**Fecha fin:**01/11/17
Descripción: Como desarrollador necesito implementar la interfaz en la que se presentarán los horarios de una cooperativa de transporte que pertenecen a una ruta.		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_21	Verificar que el diseño de la interfaz siga los diseños de los bosquejos provistos por el cliente.	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 21.

PRUEBA DE ACEPTACIÓN	
Id: PA_21	**Nombre:** Verificar que el diseño de la interfaz siga los diseños de los bosquejos provistos por el cliente.
Tarea de ingeniería: TI_01 Diseño de interfaz para consulta de horarios por cooperativa de transporte	
Descripción: La interfaz debe seguir los bosquejos provistos por el cliente basándose en colores y formatos de texto.	
Responsable: Wilmer Barrera	**Fecha:** 01/11/17
Condición de ejecución: - Tener bosquejo en el que se basará dicha interfaz - Tener instalada el IDE de desarrollo a utilizar - Tener instalado un emulador para ejecutar aplicación	
Pasos de ejecución: - Verificar que la lista de componentes de interfaz sean los maquetados en los bosquejos - Verificar que el color siga el bosquejo provisto - Verificar que los textos sigan el formato definido en el bosquejo de la app	
Resultado esperado: La interfaz gráfica cumple con los diseños del bosquejo provisto por el cliente	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla historia de usuario 03.

HISTORIA DE USUARIO		
Id: HU_03	**Nombre:** Implementación de servicios web Geocoding API de Google Maps a través de petición GET	
Descripción: Como desarrollador necesito crear una funcionalidad que me permita realizar consumo del servicio web geocoding de Google Maps a través de peticiones http por método get. El mismo que servirá para el servicio de localización		
Usuario: Wilmer Barrera	**Sprint:** 3	
Fecha inicio: 02/11/17	**Fecha fin:** 08/11/17	**Esfuerzo:** 25

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Implementación de funcionalidad para realizar peticiones de consumo de web service a través de método get
TI_02	Implementación de petición web en segundo plano

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 03.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 3
Nombre de historia usuario: HU_03 Implementación de servicios web Geocoding API de Google Maps a través de petición GET		
Nombre tarea: Implementación de funcionalidad para realizar peticiones de consumo de web service a través de método get.		
Fecha inicio: 02/11/17		**Fecha fin:**06/11/17
Descripción: Como desarrollador necesito implementar una funcionalidad en la que me permita realizar el consumo de servicios web a través de una petición http método get		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_22	Verificar que haya respuesta del consumo de servicios web de Google Maps
PA_23	Verificar que consumo de servicio web no fue exitoso
PA_24	Verificar que consumo de servicio web fue exitoso
PA_25	Verificar que consumo de servicio web fue exitoso, pero sin resultados
PA_26	Verificar respuesta de servicio web contiene lista de resultados
PA_27	Verificar que la conexión al servicio web fue aceptada
PA_28	Verificar que la conexión al servicio web fue rechazada

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 22.

PRUEBA DE ACEPTACIÓN	
Id: PA_22	**Nombre:** Verificar que haya respuesta del consumo de servicios web de Google Maps
Tarea de ingeniería: TI_01 Implementación de funcionalidad para realizar peticiones de consumo de web service a través de método get.	
Descripción: cuando se necesite realizar una geolocalización se deberá verificar que la respuesta del servicio web geocoding tenga una respuesta	
Responsable: Wilmer Barrera	**Fecha:** 06/11/17
Condición de ejecución: - Tener realizada la conexión al servicio web - Tener instalada la última versión de la app en el emulador - Tener conexión a internet	
Pasos de ejecución: - Realizar una petición al servicio web geocoding enviando los parámetros de latitud y longitud - Ejecutar la app en el emulador - Verificar que exista conexión a internet - Verificar el mensaje de respuesta del servicio web de geocoding	
Resultado esperado: el consumo del servicio web en base a la latitud y longitud, y verificar cuando no hay alguna respuesta de este consumo, la respuesta se mostrará en la pantalla como un mensaje vacío, dando a entender que no hay respuesta en tal consumo.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 23.

PRUEBA DE ACEPTACIÓN	
Id: PA_23	**Nombre:** Verificar que consumo de servicio web no fue exitoso
Tarea de ingeniería: TI_01 Implementación de funcionalidad para realizar peticiones de consumo de web service a través de método get.	
Descripción: cuando se realiza el consumo del servicio web y este es fallido entonces a aplicación deberá actuar ante este caso.	
Responsable: Wilmer Barrera	**Fecha:** 06/11/17
Condición de ejecución: - Tener instalada la última versión de la app - Tener desarrollado la conexión con los servicios web	

- Tener conexión a internet
Pasos de ejecución: - Ingresar datos errores de longitud y latitud para realizar consumo de servicio web geocoding, en este caso se enviará los datos de latitud y longitud vacíos. - Ejecutar el emulador - Ejecutar app sobre el emulador - Verificar tipo de respuesta que se presenta en pantalla
Resultado esperado: cuando no se envían datos correctos para que el consumo de servicio web nos dé respuesta, y esta petición es incorrecta entonces nos mostrará un mensaje dando a conocer que dicha petición no fue exitosa.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 24.

PRUEBA DE ACEPTACIÓN	
Id: PA_24	**Nombre:** Verificar que consumo de servicio web fue exitoso
Tarea de ingeniería: TI_01 Implementación de funcionalidad para realizar peticiones de consumo de web service a través de método get.	
Descripción: cuando se realice una petición de consumo de servicio web al api de geocoding se necesitará saber si dicha petición trae una respuesta exitosa.	
Responsable: Wilmer Barrera	**Fecha:** 06/11/17
Condición de ejecución: - Tener acceso a internet - Tener instala la última versión de la app - tener desarrollada la conexión al servicio web de geocoding	
Pasos de ejecución: - ingresar valores la latitud y longitud en la petición a geocoding - ejecutar el emulador con la app - verificar que exista conexión a internet - verificar el tipo de mensaje que devuelve la petición realizada	
Resultado esperado: al realizar una petición a geocoding enviando los parámetros correctos de latitud y longitud, esto nos trae una respuesta con un estad OK, el cual se presenta en pantalla.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 5s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 25.

PRUEBA DE ACEPTACIÓN	
Id: PA_25	**Nombre:** Verificar que consumo de servicio web fue exitoso, pero sin resultados
Tarea de ingeniería: TI_01 Implementación de funcionalidad para realizar peticiones de consumo de web service a través de método get.	
Descripción: cuando se consume un servicio web, al ser este exitoso, pero no traer resultados encontrados el sistema debe actuar sobre este caso.	
Responsable: Wilmer Barrera	**Fecha:** 06/11/17
Condición de ejecución: - Tener realizada la conexión al servicio web geocoding - Tener instalada la última versión de la app en el emulador - Tener una conexión a internet.	
Pasos de ejecución: - Ingresar datos de los cuales no existen resultados registrados por el servicio geocoding - Ejecutar la pp en el emulador - Verificar que exista conexión a Internet - Verificar mensaje que se muestra en la pantalla, como la respuesta que trae la petición del servicio web.	
Resultado esperado: cuando se consume un servicio web y no se obtiene resultados de acuerdo a los datos de latitud y longitud enviados, este imprimirá el mensaje ZERO_RESULTS, lo cual indica que no se obtuvo resultados.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 26.

PRUEBA DE ACEPTACIÓN	
Id: PA_26	**Nombre:** Verificar respuesta de servicio web contiene lista de resultados
Tarea de ingeniería: TI_01 Implementación de funcionalidad para realizar peticiones de consumo de web service a través de método get.	
Descripción: al realizar el consumo de servicio web geocoding la petición debe traer una lista de respuestas las cuales serán procesadas para presentar en la app móvil.	
Responsable: Wilmer Barrera	**Fecha:** 06/11/17
Condición de ejecución: - Tener acceso a internet - Tener instalada la última versión de la app - Tener desarrollado la conexión al servicio web geocoding	

Pasos de ejecución: - Ingresar datos correctos de latitud y longitud para la petición del servicio web - Ejecutar el emulador con la app - Verificar que en el proceso de extracción de resultados no sea un valor nulo	
Resultado esperado: cuando se consume el servicio web geocoding este nos dará una lista de respuesta de acuerdo a los datos enviados, esta respuesta se almacenará en objetos. En caso de que haya la lista de resultados estos objetos no serán nulos y la aplicación se seguirá ejecutando normalmente	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 5s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 27.

PRUEBA DE ACEPTACIÓN	
Id: PA_27	**Nombre:** Verificar que la conexión al servicio web fue aceptada
Tarea de ingeniería: TI_01 Implementación de funcionalidad para realizar peticiones de consumo de web service a través de método get.	
Descripción: para realizar el consumo del servicio web geocoding se debe hacer la conexión al mismo, por ende, se debe verificar si la petición a dicho servicio web fue aceptada	
Responsable: Wilmer Barrera	**Fecha:** 06/11/17
Condición de ejecución: - Verificar que exista conexión a internet - Tener los permisos dentro de la aplicación para tener acceso a internet - Tener desarrollado el archivo de conexión al servicio web - Tener instalada la última versión de la app en el emulador.	
Pasos de ejecución: - Ingresar datos de latitud y longitud en la petición de consumo del servicio web - Ejecutar la app en el emulador - Verificar que exista internet - Verificar presentación de mensaje de conexión aceptada	
Resultado esperado: Para poder realizar un consumo correcto del servicio web geocoding se debe hacer una petición, la cual debe ser aceptada si es hecha de manera correcta, en este caso presentará un mensaje de conexión aceptada.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 2s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 28.

PRUEBA DE ACEPTACIÓN	
Id: PA_28	**Nombre:** Verificar que la conexión al servicio web fue rechazada
Tarea de ingeniería: TI_01 Implementación de funcionalidad para realizar peticiones de consumo de web service a través de método get.	
Descripción: para realizar el consumo del servicio web geocoding se debe hacer la conexión al mismo, por ende, se debe verificar si la petición a dicho servicio web fue rechazada	
Responsable: Wilmer Barrera	**Fecha:** 06/11/17
Condición de ejecución: - Verificar que exista conexión a internet - Tener los permisos dentro de la aplicación para tener acceso a internet - Tener desarrollado el archivo de conexión al servicio web - Tener instalada la última versión de la app en el emulador.	
Pasos de ejecución: - Ingresar datos de latitud y longitud en la petición de consumo del servicio web - Ejecutar la app en el emulador - Verificar que exista internet - Verificar presentación de mensaje de conexión rechazada.	
Resultado esperado: Para poder realizar un consumo del servicio web geocoding se debe hacer una petición, la cual puede ser rechazada, en este caso presentará un mensaje de conexión rechazada.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 2s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 02 de historia de usuario 03.

TAREA DE INGENIERÍA		
Id: TI_02	**Tipo de tarea:** desarrollo	**Sprint:** 3
Nombre de historia usuario: HU_03 Implementación de servicios web Geocoding API de Google Maps a través de petición GET		
Nombre tarea: Implementación de petición web en segundo plano.		
Fecha inicio: 07/11/17		**Fecha fin:**08/11/17
Descripción: Como desarrollador necesito que el consumo del servicio web geocoding de Google Maps se ejecute en segundo plano para que el dispositivo celular no asuma que la aplicación móvil no responde.		
PRUEBAS DE ACEPTACIÓN		

Id	Nombre
PA_29	Verificar que ejecución de consumo de servicio web en segundo plano

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 29.

PRUEBA DE ACEPTACIÓN	
Id: PA_29	**Nombre:** Verificar que ejecución de consumo de servicio web en segundo plano
Tarea de ingeniería: TI_02 Implementación de petición web en segundo plano	
Descripción: Para el consumo del servicio web geocoding es necesario que la petición se ejecute en segundo plano, esto con el fin de que el sistema operativo no asuma que la aplicación se detuvo esperando una respuesta ejecutada en el hilo principal de la app	
Responsable: Wilmer Barrera	**Fecha:** 08/11/17
Condición de ejecución: - Tener creada la conexión al servicio web - Tener instalada la última versión de la app en el emulador	
Pasos de ejecución: - Ejecutar el emulador con la aplicación - Esperar 6 segundos para verificar que la aplicación no se detenga - Verificar que se pueda seguir usando la aplicación normalmente.	
Resultado esperado: al ejecutar la petición de consumo del servicio web en segundo plano este no detendrá la aplicación en caso de tardarse en traer una respuesta, así pase más de 5 segundos.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 7s

Realizado por: Wilmer B. 2018.

Sprint 4

Tabla historia de usuario 04.

HISTORIA DE USUARIO		
Id: HU_04	**Nombre:** Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps	
Descripción: Como desarrollador necesito implementar la funcionalidad para filtrado de datos de respuesta del servicio web de geocoding de Google Maps, para tener solo respuestas que son necesarias, descartando otras innecesarias.		
Usuario: Wilmer Barrera	**Sprint:** 4	
Fecha inicio: 13/11/17	**Fecha fin:** 15/11/17	**Esfuerzo:** 15

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Implementación de funcionalidad para recolectar resultados de petición a servicio web geocoding
TI_02	Implementación de funcionalidad para filtrar resultado devuelvo de servicio web geocoding
TI_03	Implementación de funcionalidad para formatear cadenas de direcciones
TI_04	Implementación de funcionalidad para encontrar coincidencias de provincias del Ecuador con el listado de resultados devuelto por el servicio geocoding
TI_05	Implementación de funcionalidad para encontrar coincidencias de ciudades del Ecuador con el listado de resultados devuelto por el servicio geocoding
TI_06	Implementación de funcionalidad para obtener la distancia entre dos coordenadas geográficas.
TI_07	Implementar funcionalidad para obtener ciudad más cercana de acuerdo a una dirección específica

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 04.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 4
Nombre de historia usuario: HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps		
Nombre tarea: Implementación de funcionalidad para recolectar resultados de petición a servicio web geocoding.		
Fecha inicio: 13/11/17		**Fecha fin:**13/11/17
Descripción: Como desarrollador necesito implementar la funcionalidad para obtener o extraer los resultados de petición de servicio web geocoding		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_30	Verificar que la lista de resultados no esté vacía	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 30.

PRUEBA DE ACEPTACIÓN	
Id: PA_30	**Nombre:** Verificar que la lista de resultados no esté vacía
Tarea de ingeniería: TI_01 Implementación de funcionalidad para recolectar resultados de petición a servicio web geocoding.	
Descripción: cuando se ha realizado una petición de consumo de servicio web geocoding y no existe respuesta, ante este este caso la aplicación deberá reaccionar ante esta.	
Responsable: Wilmer Barrera	**Fecha:** 13/11/17
Condición de ejecución: - Tener desarrollado la conexión al servicio web - Verificar conexión a internet - Tener instalada la última versión de la app en el emulador	
Pasos de ejecución: - Enviar datos de latitud y longitud en petición a geocoding - Ejecutar la aplicación en el emulador - Verificar que la lista de respuesta sea nula	
Resultado esperado: al no existir una lista de respuestas traídas desde el servicio de geocoding por ende existirá una lista nula, sin datos esto se mostrará a través de un mensaje.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 02 de historia de usuario 04.

TAREA DE INGENIERÍA		
Id: TI_02	**Tipo de tarea:** desarrollo	**Sprint:** 4
Nombre de historia usuario: HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps		
Nombre tarea: Implementación de funcionalidad para filtrar resultado devuelvo de servicio web geocoding.		
Fecha inicio: 13/11/17		**Fecha fin:**13/11/17
Descripción: Como desarrollador necesito filtrar los resultados devueltos del servicio web geocoding para tener solo los necesarios.		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_31	Verificar que existe lista de direcciones filtradas	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 31.

PRUEBA DE ACEPTACIÓN	
Id: PA_31	**Nombre:** Verificar que existe lista de direcciones filtradas
Tarea de ingeniería: TI_02 Implementación de funcionalidad para filtrar resultado devuelvo de servicio web geocoding.	
Descripción: Cuando se realiza una petición al servicio web geocoding este debe devolvernos una respuesta, la cual será, procesada posteriormente.	
Responsable: Wilmer Barrera	**Fecha:** 13/11/17
Condición de ejecución: - Tener una conexión a internet - Tener desarrollada la conexión al servicio web geocoding - Tener instalada la última versión de la app	
Pasos de ejecución: - Ingresar daos de latitud y longitud en la app - Ejecutar la app en el emulador - Verificar que la lista de direcciones o respuestas sea diferente de un valor nulo	
Resultado esperado: cuando se extraiga la lista de resultados estos deben ser diferentes de un valor nulo, imprimiendo así el número de elementos de la lista.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 03 de historia de usuario 04.

TAREA DE INGENIERÍA		
Id: TI_03	**Tipo de tarea:** desarrollo	**Sprint:** 4
Nombre de historia usuario: HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps		
Nombre tarea: Implementación de funcionalidad para formatear cadenas de direcciones.		
Fecha inicio: 14/11/17		**Fecha fin:**14/11/17
Descripción: Con el fin de mostrar una lista de direcciones entendibles para el usuario y tener una presentación de acuerdo a los requerimientos del cliente, se formateará las direcciones para presentarlas		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_32	Verificar lista de direcciones fueron formateadas exitosamente	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 32.

PRUEBA DE ACEPTACIÓN	
Id: PA_32	**Nombre:** Verificar lista de direcciones fueron formateadas exitosamente
Tarea de ingeniería: TI_03 Implementación de funcionalidad para formatear cadenas de direcciones	
Descripción: al tener un resultado desde el servicio web geocoding esta lista de resultados deben ser procesados, para que se pueda obtener solamente la información útil, descartando datos innecesarios y repetidos.	
Responsable: Wilmer Barrera	**Fecha:** 14/11/17
Condición de ejecución: - Tener desarrollada la conexión al servicio web geocoding - Tener instalada la última versión de la app en el emulador - Tener una conexión a internet	
Pasos de ejecución: - Agregar datos obligatorios para consumo de servicio web geocoding, estos datos son latitud y longitud - Ejecutar app móvil desde el emulador. - Verificar que lista de respuesta sea la óptima, necesaria y no repetida	
Resultado esperado: Cuando se manda a formatear una lista de direcciones, entonces está se guardará en una lista de strings evitando la redundancia de las direcciones para posteriormente seguir a otro proceso de filtrado de datos. Esta información se confirmará a través de la impresión de la lista de resultados formateados en modo consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 5s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 04 de historia de usuario 04.

TAREA DE INGENIERÍA		
Id: TI_04	**Tipo de tarea:** desarrollo	**Sprint:** 4
Nombre de historia usuario: HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps		
Nombre tarea: Implementación de funcionalidad para encontrar coincidencias de provincias del Ecuador con el listado de resultados devuelto por el servicio geocoding.		
Fecha inicio: 14/11/17		**Fecha fin:** 14/11/17
Descripción:		

Dentro del conjunto de respuesta que tiene la lista de direcciones se comparará con todas las provincias almacenadas en la base de datos, encontrando coincidencias de estas, la cual servirá para el proceso de localización del punto donde se encuentra el dispositivo.

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_33	Verificar que cargaron lista de provincias desde la base de datos
PA_34	Verificar que tabla provincias no se encuentre vacía
PA_35	Verificar que se encontró coincidencia entre una provincia y el listado de direcciones filtradas
PA_36	Verificar que no se encontró coincidencia entre una provincia y el listado de direcciones filtradas

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 33.

PRUEBA DE ACEPTACIÓN	
Id: PA_33	**Nombre:** Verificar que cargaron lista de provincias desde la base de datos
Tarea de ingeniería: TI_04 Implementación de funcionalidad para encontrar coincidencias de provincias del Ecuador con el listado de resultados devuelto por el servicio geocoding.	
Descripción: Es necesario poder almacenar los datos de las provincias en la base de datos local para posteriores consultas y procesos que se relacionan a estos.	
Responsable: Wilmer Barrera	**Fecha:** 14/11/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla de datos PROVINCIAS - Tener instalada la última versión de la app - Tener ingresado datos dentro de la tabla PROVINCIAS	
Pasos de ejecución: - Ejecutar la app desde el emulador - Ejecutar parte de la app donde se prueba la extracción de todas las provincias desde la base de datos local. - Verificar que se extraigan las 24 provincias almacenadas en la tabla PROVINCIAS	
Resultado esperado: Cuando se manda a cargar la lista de provincias del Ecuador desde la base de datos local deberá extraer una lista con 24 elementos, correspondientes a todas las provincias del Ecuador. Este número se dará a conocer imprimiéndose en la aplicación en modo consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 34.

PRUEBA DE ACEPTACIÓN	
Id: PA_34	**Nombre:** Verificar que tabla provincias se encuentre vacía
Tarea de ingeniería: TI_04 Implementación de funcionalidad para encontrar coincidencias de provincias del Ecuador con el listado de resultados devuelto por el servicio geocoding.	
Descripción: Cuando se realiza una consulta a la base de datos local queriendo cargar todos los datos de las provincias, esta tabla puede encontrarse vacía, por lo que se necesita controlar si pasa este tipo de situaciones.	
Responsable: Wilmer Barrera	**Fecha:** 14/11/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla PROVINCIAS - Tener la tabla PROVINCIAS vacía - Tener instalada la última versión de la app en el emulador	
Pasos de ejecución: - Ejecutar la app en el emulador - Verificar que se ejecute el proceso de hacer una consulta a la base de datos, con el fin de obtener la lista de provincias del Ecuador. - Verificar la cantidad de elementos devueltos por la consulta	
Resultado esperado: Cuando se intenta extraer la lista de provincias desde la base de datos local, y no existen aún provincias registradas entonces devolverá una lista con 0 elementos, este resultado se mostrará en la consola de la app.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 35.

PRUEBA DE ACEPTACIÓN	
Id: PA_35	**Nombre:** Verificar que se encontró coincidencia entre una provincia y el listado de direcciones filtradas
Tarea de ingeniería: TI_04 Implementación de funcionalidad para encontrar coincidencias de provincias del Ecuador con el listado de resultados devuelto por el servicio geocoding.	
Descripción: Para poder llevar a cabo la localización es necesario procesar la lista de resultados extraídos desde el servicio geocoding, este proceso se debe llevar a cabo encontrando coincidencias entre la lista de provincias de la base de datos local y la lista de direcciones suministradas por el servicio geocoding.	
Responsable: Wilmer Barrera	**Fecha:** 14/11/17

Condición de ejecución: - Tener creada la base de datos - Tener datos en la tabla PROVINCIAS - Tener instalada la última versión de la app en el emulador - Tener acceso a internet en el emulador
Pasos de ejecución: - Ejecutar la app en el emulador - Verificar si se encontró coincidencia de una de las provincias con la lista de direcciones extraídas como resultado. - Verificar que se imprima en consola esta coincidencia de resultados.
Resultado esperado: Cuando se tiene una lista de direcciones filtradas y formateadas, entonces se cotejará la lista de direcciones con la lista de provincias, del cual al encontrará una coincidencia devolverá un id de provincia, la cual será presentada en la consola.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 36.

PRUEBA DE ACEPTACIÓN	
Id: PA_36	**Nombre:** Verificar que no se encontró coincidencia entre una provincia y el listado de direcciones filtradas
Tarea de ingeniería: TI_04 Implementación de funcionalidad para encontrar coincidencias de provincias del Ecuador con el listado de resultados devuelto por el servicio geocoding.	
Descripción: La lista de resultados extraídos desde el servicio geocoding al ser comparados con la lista de provincias de la base de datos local y no encontrar coincidencias el sistema debe actuar ante este caso.	
Responsable: Wilmer Barrera	**Fecha:** 14/11/17
Condición de ejecución: - Tener creada la base de datos - Tener datos en la tabla PROVINCIAS - Tener instalada la última versión de la app en el emulador - Tener acceso a internet en el emulador	
Pasos de ejecución: - Ejecutar la app en el emulador - Verificar que no se encontró coincidencia de ninguna de las provincias con la lista de direcciones extraídas como resultado.	

- Verificar que se imprima en consola el resultado correspondiente cuando no existe coincidencia de provincias.
Resultado esperado: Cuando se busca coincidencias de provincias entre la lista de direcciones y la lista de provincias extraída de la base de datos, entonces no se encuentra ningún id de provincia por lo que devolverá un valor de -1, el mismo que será mostrado en consola.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 05 de historia de usuario 04.

TAREA DE INGENIERÍA		
Id: TI_05	**Tipo de tarea:** desarrollo	**Sprint:** 4
Nombre de historia usuario: HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps		
Nombre tarea: Implementación de funcionalidad para encontrar coincidencias de ciudades del Ecuador con el listado de resultados devuelto por el servicio geocoding.		
Fecha inicio: 14/11/17		**Fecha fin:** 14/11/17
Descripción: Dentro del conjunto de respuesta que tiene la lista de direcciones se comparará con todas las ciudades de una provincia almacenadas en la base de datos, encontrando coincidencias de estas, la cual servirá para el proceso de localización del punto donde se encuentra el dispositivo.		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_37	Verificar que se cargue lista de ciudades de una provincia
PA_38	Verificar que lista de ciudades de una provincia no esté vacía
PA_39	Verificar cuando se encontró coincidencias de ciudades con la lista de direcciones filtradas
PA_40	Verificar cuando no se encontró coincidencias de ciudades con la lista de direcciones filtradas

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 37.

PRUEBA DE ACEPTACIÓN	
Id: PA_37	**Nombre:** Verificar que se cargue lista de ciudades de una provincia
Tarea de ingeniería: TI_05 Implementación de funcionalidad para encontrar coincidencias de ciudades del Ecuador con el listado de resultados devuelto por el servicio geocoding.	
Descripción: Es necesario poder almacenar los datos de las ciudades que corresponden a una provincia en la base de datos local para posteriores consultas y procesos que se relacionan a estos.	
Responsable: Wilmer Barrera	**Fecha:** 14/11/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla de datos CIUDADES - Tener instalada la última versión de la app - Tener ingresado datos dentro de la tabla CIUDADES	
Pasos de ejecución: - Ejecutar la app desde el emulador - Ejecutar parte de la app donde se prueba la extracción de todas las ciudades de una provincia desde la base de datos local. - Verificar que se extraigan todas las ciudades de unas provincias almacenadas en la tabla CIUDADES.	
Resultado esperado: Cuando se manda a cargar la lista de ciudades de una provincia del Ecuador desde la base de datos local deberá extraer una lista con todas las ciudades correspondientes a todas las provincias del Ecuador. Este número se dará a conocer imprimiéndose en la aplicación en modo consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 38.

PRUEBA DE ACEPTACIÓN	
Id: PA_38	**Nombre:** Verificar que lista de ciudades de una provincia esté vacía
Tarea de ingeniería: TI_05 Implementación de funcionalidad para encontrar coincidencias de ciudades del Ecuador con el listado de resultados devuelto por el servicio geocoding.	
Descripción: Cuando se realiza una consulta a la base de datos local queriendo cargar todos los datos de las ciudades de una provincia, esta tabla puede encontrarse vacía, por lo que se necesita controlar si pasa este tipo de situaciones.	
Responsable: Wilmer Barrera	**Fecha:** 14/11/17

Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla CIUDADES - Tener la tabla CIUDADES vacía - Tener instalada la última versión de la app en el emulador
Pasos de ejecución: - Ejecutar la app en el emulador - Verificar que se ejecute el proceso de hacer una consulta a la base de datos, con el fin de obtener la lista de ciudades de una provincia del Ecuador. - Verificar la cantidad de elementos devueltos por la consulta
Resultado esperado: Cuando se intenta extraer la lista de ciudades de una provincia desde la base de datos local, y no existen aún ciudades registradas entonces devolverá una lista con 0 elementos, este resultado se mostrará en la consola de la app.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 39.

PRUEBA DE ACEPTACIÓN	
Id: PA_39	**Nombre:** Verificar cuando se encontró coincidencias de ciudades con la lista de direcciones filtradas
Tarea de ingeniería: TI_05 Implementación de funcionalidad para encontrar coincidencias de ciudades del Ecuador con el listado de resultados devuelto por el servicio geocoding.	
Descripción: Para poder llevar a cabo la localización es necesario procesar la lista de resultados extraídos desde el servicio geocoding, este proceso se debe llevar a cabo encontrando coincidencias entre la lista de ciudades que tiene una provincia de la base de datos local y la lista de direcciones suministradas por el servicio geocoding.	
Responsable: Wilmer Barrera	**Fecha:** 14/11/17
Condición de ejecución: - Tener creada la base de CIUDADES - Tener datos en la tabla CIUDADES - Tener instalada la última versión de la app en el emulador - Tener acceso a internet en el emulador	
Pasos de ejecución: - Ejecutar la app en el emulador - Verificar si se encontró coincidencia de una de las ciudades pertenecientes a una provincia con la lista de direcciones extraídas como resultado.	

- Verificar que se imprima en consola esta coincidencia de resultados.
Resultado esperado: Cuando se tiene una lista de direcciones filtradas, entonces se cotejará la lista de direcciones con la lista de ciudades, del cual al encontrará una coincidencia devolverá la ciudad que coincidió, la cual será presentada en la consola.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:**3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 40.

PRUEBA DE ACEPTACIÓN	
Id: PA_40	**Nombre:** Verificar cuando no se encontró coincidencias de ciudades con la lista de direcciones filtradas
Tarea de ingeniería: TI_05 Implementación de funcionalidad para encontrar coincidencias de ciudades del Ecuador con el listado de resultados devuelto por el servicio geocoding.	
Descripción: La lista de resultados extraídos desde el servicio geocoding al ser comparados con la lista de ciudades de la base de datos local y no encontrar coincidencias el sistema debe actuar ante este caso.	
Responsable: Wilmer Barrera	**Fecha:** 14/11/17
Condición de ejecución: - Tener creada la base de datos - Tener datos en la tabla CIUDADES - Tener instalada la última versión de la app en el emulador - Tener acceso a internet en el emulador	
Pasos de ejecución: - Ejecutar la app en el emulador - Verificar que no se encontró coincidencia de ninguna de las ciudades de una provincia con la lista de direcciones extraídas como resultado. - Verificar que se imprima en consola el resultado correspondiente cuando no existe coincidencia de ciudades.	
Resultado esperado: Cuando se busca coincidencias de ciudades de una provincia entre la lista de direcciones y la lista de ciudades extraída de la base de datos, entonces no se encuentra ningún resultado, dando así un valor nulo como resultado, el mismo que será mostrado en consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 2s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 06 de historia de usuario 04.

TAREA DE INGENIERÍA		
Id: TI_06	**Tipo de tarea:** desarrollo	**Sprint:** 4
Nombre de historia usuario: HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps		
Nombre tarea: Implementación de funcionalidad para obtener la distancia entre dos coordenadas geográficas.		
Fecha inicio: 15/11/17		**Fecha fin:** 15/11/17
Descripción: Con el fin de obtener una ciudad cercana una de otra se necesita obtener la distancia entre dos puntos geográficos esto siguiendo el método de esfera.		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_41	Verificación de conversión cantidad normal a radianes	
PA_42	Verificar distancia entre dos coordenadas geográficas	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 41.

PRUEBA DE ACEPTACIÓN	
Id: PA_41	**Nombre:** Verificación de conversión cantidad normal a radianes
Tarea de ingeniería: TI_06 Implementación de funcionalidad para obtener la distancia entre dos coordenadas geográficas.	
Descripción: para poder llevar a cabo la obtención de la distancia entre dos puntos de manera esférica uno de los procesos es la transformación de una cantidad numérica en radianes.	
Responsable: Wilmer Barrera	**Fecha:** 15/11/17
Condición de ejecución: - Tener instalado un emulador - Tener instalada la última versión de la app en el emulador - Tener implementada a funcionalidad para conversión de un número a radianes	
Pasos de ejecución: - Ingresar una cantidad numérica para probar la conversión de un número a radianes - Ejecutar la aplicación desde el emulador. - Verificar que se imprima el resultado correcto en radianes en consola	
Resultado esperado: Con la obtención la distancia entre dos puntos esféricamente en base a coordenadas GPS, se necesita pasar dicho resultado a radianes el cual se imprime en consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 5s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 42.

PRUEBA DE ACEPTACIÓN	
Id: PA_42	**Nombre:** Verificar distancia entre dos coordenadas geográficas
Tarea de ingeniería: TI_06 Implementación de funcionalidad para obtener la distancia entre dos coordenadas geográficas.	
Descripción: con el fin de obtener la distancia esférica entre dos puntos para obtener una ciudad más cercana a través de parámetros de latitud y longitud.	
Responsable: Wilmer Barrera	**Fecha:** 15/11/17
Condición de ejecución: - Tener instalado un emulador - Tener instalada la última versión de la app - Tener desarrollada la funcionalidad para calcular la distancia entre dos puntos esféricamente.	
Pasos de ejecución: - Ingresar los dos puntos que se obtendrá la distancia, los puntos obligatorios son dos coordenadas de latitud y 2 de longitud. - Ejecutar la aplicación desde el emulador - Verificar que la distancia entre los dos puntos esféricamente sea correcta	
Resultado esperado: Para buscar la ciudad más cercana en base a coordenadas GPS se saca la distancia entre dos puntos esféricamente, en este proceso nos da como resultado la distancia más aproximada entre dichos puntos. Este resultado será presentado en modo consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 07 de historia de usuario 04.

TAREA DE INGENIERÍA		
Id: TI_07	**Tipo de tarea:** desarrollo	**Sprint:** 4
Nombre de historia usuario: HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps		
Nombre tarea: Implementar funcionalidad para obtener ciudad más cercana de acuerdo a una dirección específica.		
Fecha inicio: 15/11/17		**Fecha fin:** 15/11/17
Descripción: Partiendo de la ubicación del dispositivo celular se implementará una funcionalidad en la cual se obtendrá la distancia entre la lista de ciudades de una provincia y el punto en el que se		

encuentra el dispositivo, encontrando así la ciudad más cercana donde puede tener el servicio de transporte.

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_43	Verificar que exista índice menor de una lista de ciudades cercana a una dirección específica
PA_44	Verificar que se cargue toda la lista de ciudades de la base de datos local
PA_45	Verificar que lista de ciudades extraídas de la base de datos no esté vacía
PA_46	Verificar si existe una ciudad cercana
PA_47	Verificar cuando no existe una ciudad cercana
PA_48	Verificar existencia de una ciudad específica dentro de la tabla CIUDADES
PA_49	Verificar cuando no existe una ciudad específica dentro de la tabla CIUDADES

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 43.

PRUEBA DE ACEPTACIÓN	
Id: PA_43	**Nombre:** Verificar que exista índice menor de una lista de ciudades cercana a una dirección específica
Tarea de ingeniería: TI_07 Implementar funcionalidad para obtener ciudad más cercana de acuerdo a una dirección específica.	
Descripción: para poder encontrar una ciudad cercana a un punto basado en la latitud y longitud, se debe buscar la ciudad cercana entre una lista de ciudades que están almacenados en la base de datos local.	
Responsable: Wilmer Barrera	**Fecha:** 15/11/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla ciudades y tener almacenado datos - Tener instalada la última versión de la app - Tener instalado un emulador	
Pasos de ejecución: - Insertar el punto de latitud y longitud en que se basará la ciudad más cercana - Ejecutar la app en el emulador - Verificar que presente un valor numérico en modo consola de la app	

Resultado esperado: cuando se encuentre una ciudad cercana dentro de una lista de ciudades este proceso devolverá un valor entero que representa dicha ciudad, este resultado se mostrará en la consola de la app móvil.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 44.

PRUEBA DE ACEPTACIÓN	
Id: PA_44	**Nombre:** Verificar que se cargue toda la lista de ciudades de la base de datos local
Tarea de ingeniería: TI_07 Implementar funcionalidad para obtener ciudad más cercana de acuerdo a una dirección específica.	
Descripción: para poder obtener la ciudad más cercana en base a un punto geográfico debemos extraer la lista de ciudades de la base de datos para sacar la distancia entre dos puntos entre cada una de ellas y así obtener la ciudad más cercana.	
Responsable: Wilmer Barrera	**Fecha:** 15/11/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla CIUDADES con datos almacenados - Tener instalada la última versión de la app	
Pasos de ejecución: - Ejecutar la aplicación desde el emulador - Verificar que se imprima el resultado de la cantidad de elementos extraídos de la base de datos local - Verificar que se imprima el resultado en modo consola	
Resultado esperado: al extraer todas las ciudades de la tabla CIUDADES este proceso debe imprimir un número entero mayor a cero, lo que significa que ese es el número de elementos extraídos de la base de datos local.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 45.

PRUEBA DE ACEPTACIÓN	
Id: PA_45	**Nombre:** Verificar que lista de ciudades extraídas de la base de datos esté vacía
Tarea de ingeniería: TI_07 Implementar funcionalidad para obtener ciudad más cercana de acuerdo a una dirección específica.	
Descripción: cuando se intente extraer la lista de ciudades desde la base de datos y esta se encuentre vacía el sistema deberá actuar antes esta situación.	
Responsable: Wilmer Barrera	**Fecha:** 15/11/17
Condición de ejecución: - Tener creada la base de datos - Tener un emulador instalado - Tener instalada la última versión de la app - Tener la tabla CIUDADES vacía	
Pasos de ejecución: - Ejecutar la app desde el emulador - Verificar que la aplicación muestre el resultado de la consulta en la consola	
Resultado esperado: cuando se realice una consulta de todas las ciudades almacenadas en la base de datos y está se encuentre vacía, entonces se imprimirá el número cero, que corresponde a que cero ciudades fueron encontradas.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 2s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 46.

PRUEBA DE ACEPTACIÓN	
Id: PA_46	**Nombre:** Verificar si existe una ciudad cercana
Tarea de ingeniería: TI_07 Implementar funcionalidad para obtener ciudad más cercana de acuerdo a una dirección específica.	
Descripción: cuando se necesite buscar una ciudad cercana basándose en un punto geográfico entonces se deberá buscar con la lista de ciudades almacenadas en la base de datos local.	
Responsable: Wilmer Barrera	**Fecha:** 15/11/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la funcionalidad para encontrar la ciudad más cercana - Tener la tabla CIUDADES con 3 registros. - Tener instalada la última versión de la app instalada.	
Pasos de ejecución:	

- Agregar los parámetros de latitud y longitud en los que se basará la ciudad más cercana - Ejecutar la app en el emulador - Verificar que se imprima la respuesta cuando se encuentre la ciudad más cercana
Resultado esperado: cuando se encuentre la ciudad más cercana en base a un punto geográfico este resultado será mostrado en la pantalla, en el cual estará la ciudad que cumple con esta condición.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** 73s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 47.

PRUEBA DE ACEPTACIÓN	
Id: PA_47	**Nombre:** Verificar cuando no existe una ciudad cercana
Tarea de ingeniería: TI_07 Implementar funcionalidad para obtener ciudad más cercana de acuerdo a una dirección específica.	
Descripción: cuando no se encuentra una ciudad cercana en base a un punto geográfico de una ciudad entonces el sistema debe reaccionar de alguna manera ante esta situación.	
Responsable: Wilmer Barrera	**Fecha:** 15/11/17
Condición de ejecución: - Tener creada la base de datos - Tener datos en la tabla CIUDADES - Tener instalada la última versión de la app	
Pasos de ejecución: - Ingresar las coordenadas de latitud y longitud como punto base de localización de la ciudad más cercana. - Ejecutar la app móvil en el emulador - Verificar el resultado que arroja el proceso de buscar una ciudad más cercana	
Resultado esperado: cuando no se encuentre una ciudad cercana este proceso devolverá como resultado un valor booleano falso, y este resultado será presentado en el modo consola de la aplicación.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 2s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 48.

PRUEBA DE ACEPTACIÓN	
Id: PA_48	**Nombre:** Verificar existencia de una ciudad específica dentro de la tabla CIUDADES
Tarea de ingeniería: TI_07 Implementar funcionalidad para obtener ciudad más cercana de acuerdo a una dirección específica.	
Descripción: con el objetivo de buscar la ciudad más cercana se necesita verificar si existe una ciudad determinada dentro de la base de datos local.	
Responsable: Wilmer Barrera	**Fecha:** 15/11/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla CIUDADES con datos almacenados - Tener instalado un emulador - Tener instalada la última versión de la app	
Pasos de ejecución: - Asignar el parámetro de búsqueda de una ciudad específica - Ejecutar la app en el emulador - Verificar que la respuesta que muestra la consola.	
Resultado esperado: cuando **se** hace una consulta a la base de datos, con el fin de verificar si una ciudad determinada existe y la respuesta es positiva entonces este proceso mostrará en consola un valor booleano verdadero.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 49.

PRUEBA DE ACEPTACIÓN	
Id: PA_49	**Nombre:** Verificar cuando no existe una ciudad específica dentro de la tabla CIUDADES
Tarea de ingeniería: TI_07 Implementar funcionalidad para obtener ciudad más cercana de acuerdo a una dirección específica.	
Descripción: con el objetivo de buscar la ciudad más cercana se necesita verificar cuando no existe una ciudad determinada dentro de la base de datos local.	
Responsable: Wilmer Barrera	**Fecha:** 15/11/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla CIUDADES con datos almacenados	

- Tener instalado un emulador - Tener instalada la última versión de la app
Pasos de ejecución: - Asignar el parámetro de búsqueda de una ciudad específica - Ejecutar la app en el emulador - Verificar que la respuesta que muestra la consola.
Resultado esperado: cuando **se** hace una consulta a la base de datos, con el fin de buscar una ciudad determinada y esta no existe, la respuesta es negativa entonces este proceso mostrará en consola un valor booleano falso.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla historia de usuario 05.

HISTORIA DE USUARIO		
Id: HU_05	**Nombre:** Implementación de interfaz para consulta de horarios en base a ciudad origen y destino	
Descripción: Como desarrollador necesito implementar la interfaz en donde se mostrará la información de los horarios que se busca en base a una ciudad origen y destino, esto de acuerdo con los requerimientos del cliente.		
Usuario: Wilmer Barrera	**Sprint:** 4	
Fecha inicio: 16/11/17	**Fecha fin:** 16/11/17	**Esfuerzo:** 5

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Diseño de interfaz para consulta de horarios en base a ciudad origen y destino

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 05.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 4
Nombre de historia usuario: HU_05 Implementación de interfaz para consulta de horarios en base a ciudad origen y destino		
Nombre tarea: Diseño de interfaz para consulta de horarios en base a ciudad origen y destino.		
Fecha inicio: 16/11/17		**Fecha fin:**16/11/17
Descripción: Como desarrollador necesito diseñar una interfaz para presentar los datos de los horarios que se basan en rutas		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_50	Verificar que el diseño de la interfaz siga los bosquejos provistos por el cliente

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 50.

PRUEBA DE ACEPTACIÓN		
Id: PA_50	**Nombre:** Verificar que el diseño de la interfaz siga los bosquejos provistos por el cliente	
Tarea de ingeniería: TI_01 Diseño de interfaz para consulta de horarios en base a ciudad origen y destino		
Descripción: La interfaz deberá seguir el diseño de bosquejos provisto por el cliente, así como los colores y formatos de texto que se presentan en este.		
Responsable: Wilmer Barrera	**Fecha:** 16/11/17	
Condición de ejecución: - Tener instalada el IDE de desarrollo Android Studio - Tener instalado un emulador en el cual se ejecutará la app - Tener diseño de bosquejos de la interfaz		
Pasos de ejecución: - Verificar que todos los componentes de los bosquejos estén bien distribuidos en la interfaz - Verificar que los colores de la app móvil sean iguales a los de los bosquejos - Verificar que el formato de texto sea el mismo que se especifica en el bosquejo		
Resultado esperado: La interfaz sigue el diseño de los bosquejos a cabalidad, tanto en texto colores y distribución de componentes de interfaz		
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---	

Realizado por: Wilmer B. 2018.

Tabla historia de usuario 06.

HISTORIA DE USUARIO		
Id: HU_06	**Nombre:** Implementación de funcionalidad para consulta de horarios por transporte	
Descripción: Como desarrollador necesito implementar la funcionalidad en la que se pueda consultar la lista de horarios de una ruta específica de una cooperativa de transporte.		
Usuario: Wilmer Barrera	**Sprint:** 4	
Fecha inicio: 17/11/17	**Fecha fin:** 22/11/17	**Esfuerzo:** 20

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Crear funcionalidad para extracción de lista de cooperativas de transporte
TI_02	Crear funcionalidad para extracción de lista de rutas que pertenecen a una cooperativa de transporte
TI_03	Crear funcionalidad para formatear información de rutas almacenadas en lista de objetos pasando a lista de strings
TI_04	Crear funcionalidad para obtener lista de dependencias que pertenecen a una cooperativa de transporte
TI_05	Crear funcionalidad para extracción de lista de horarios en base a una ruta de una agencia de cooperativa de transporte específica.
TI_06	Crear adaptador personalizado de listview para presentar lista de horarios de una ruta por transporte

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 06.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 4
Nombre de historia usuario: HU_06 Implementación de funcionalidad para consulta de horarios por transporte		
Nombre tarea: Crear funcionalidad para extracción de lista de cooperativas de transporte.		
Fecha inicio: 17/11/17		**Fecha fin:**17/11/17
Descripción: Como desarrollador necesito crear la funcionalidad en la que se extraerán la lista de cooperativas de transporte desde la base de datos local		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_51	Verificar que se cargó lista de cooperativas de transporte desde la base de datos local	
PA_52	Verificar que lista de cooperativas de transporte esté vacía	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 51.

PRUEBA DE ACEPTACIÓN	
Id: PA_51	**Nombre:** Verificar que se cargó lista de cooperativas de transporte desde la base de datos local
Tarea de ingeniería: TI_01 Crear funcionalidad para extracción de lista de cooperativas de transporte	
Descripción: con el fin de mostrar la lista de cooperativas registradas en la app, estás se deben de cargar de manera correcta desde la base de datos.	
Responsable: Wilmer Barrera	**Fecha:** 17/11/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla COOPERATIVA - Tener datos almacenados en la tabla cooperativa - Tener instalada la última versión de la app en el emulador	
Pasos de ejecución: - Ejecutar la app desde el emulador de Android - Verificar el resultado de salida que se muestra en la consola	
Resultado esperado: cuando se realiza una consulta a la base de datos con el fin de cargar toda la lista de cooperativas y esta es exitosa nos mostrará a través de un mensaje el número de elementos que fueron extraídos desde la base de datos local.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 52.

PRUEBA DE ACEPTACIÓN	
Id: PA_52	**Nombre:** Verificar que lista de cooperativas de transporte esté vacía
Tarea de ingeniería: TI_01 Crear funcionalidad para extracción de lista de cooperativas de transporte	
Descripción: cuando se extraiga la lista de cooperativas desde la base de datos local y esta se encuentra vacía, es una situación a la que el sistema debe reaccionar	
Responsable: Wilmer Barrera	**Fecha:** 17/11/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla COOPERATIVA - Tener vacía de datos la tabla cooperativa - Tener instalada la última versión de la app en el emulador	

Pasos de ejecución: - Ejecutar la app desde el emulador de Android - Verificar el resultado de salida que se muestra en la consola	
Resultado esperado: cuando no exista datos en la tabla cooperativa, entonces la consulta nos devolverá un resultado con un número entero de 0 elementos, que se mostrará en la consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 02 de historia de usuario 06.

TAREA DE INGENIERÍA		
Id: TI_02	**Tipo de tarea:** desarrollo	**Sprint:** 4
Nombre de historia usuario: HU_06 Implementación de funcionalidad para consulta de horarios por transporte		
Nombre tarea: Crear funcionalidad para extracción de lista de rutas que pertenecen a una cooperativa de transporte.		
Fecha inicio: 17/11/17		**Fecha fin:** 17/11/17
Descripción: Como desarrollador necesito crear la funcionalidad en la que se extraerán la lista de rutas que pertenecen a una cooperativa de transporte desde la base de datos local		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_53	Verificar que se cargue lista de rutas de una cooperativa de transporte específica
PA_54	Verificar que lista de rutas de una cooperativa de transporte específica esté vacía

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 53.

PRUEBA DE ACEPTACIÓN	
Id: PA_53	**Nombre:** Verificar que se cargue lista de rutas de una cooperativa de transporte específica
Tarea de ingeniería: TI_02 Crear funcionalidad para extracción de lista de rutas que pertenecen a una cooperativa de transporte	
Descripción: para extraer una lista de rutas que cubre una cooperativa de transporte desde la base de datos local, con el fin de mostrar esta información al cliente.	
Responsable: Wilmer Barrera	**Fecha:** 17/11/17

Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla RUTAS con datos - Tener instalada la última versión de la app
Pasos de ejecución: - Agregar el parámetro de búsqueda, este parámetro es el id de la provincia - Ejecutar la aplicación desde el emulador - Verificar el mensaje de respuesta que se imprime en la consola
Resultado esperado: al extraer la lista de rutas desde la base de datos local, este proceso devolverá una lista de elementos de la cual se tomará la cantidad de elementos extraídos y se presentará en la pantalla.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 54.

PRUEBA DE ACEPTACIÓN	
Id: PA_54	**Nombre:** Verificar que lista de rutas de una cooperativa de transporte específica esté vacía
Tarea de ingeniería: TI_02 Crear funcionalidad para extracción de lista de rutas que pertenecen a una cooperativa de transporte	
Descripción: cuando se extrae la lista de rutas que cubre una cooperativa de transporte desde la base de datos local, y no existen datos esta situación debe ser controlada por el sistema.	
Responsable: Wilmer Barrera	**Fecha:** 17/11/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla RUTAS con datos - Tener instalada la última versión de la app	
Pasos de ejecución: - Agregar el parámetro de búsqueda, este parámetro es el id de la provincia - Ejecutar la aplicación desde el emulador - Verificar el mensaje de respuesta que se imprime en la consola	
Resultado esperado: al extraer la lista de rutas desde la base de datos local, este proceso devolverá una lista de elementos de la cual se tomará la cantidad de elementos extraídos en caso de no haber resultado se mostrará el número cero.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 03 de historia de usuario 06.

TAREA DE INGENIERÍA		
Id: TI_03	**Tipo de tarea:** desarrollo	**Sprint:** 4
Nombre de historia usuario: HU_06 Implementación de funcionalidad para consulta de horarios por transporte		
Nombre tarea: Crear funcionalidad para formatear información de rutas almacenadas en lista de objetos pasando a lista de strings.		
Fecha inicio: 20/11/17		**Fecha fin:**20/11/17
Descripción: Con el fin de mostrar los resultados de las rutas específicas consultadas por el usuario se formateará una lista de objetos rutas a una lista de strings.		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_55	Verificar formateo de campos de objetos a lista de strings	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 55.

PRUEBA DE ACEPTACIÓN	
Id: PA_55	**Nombre:** Verificar formateo de campos de objetos a lista de strings
Tarea de ingeniería: TI_03 Crear funcionalidad para formatear información de rutas almacenadas en lista de objetos pasando a lista de strings.	
Descripción: al recuperar lista de rutas desde la base de datos esta será almacenada en una lista de objetos que será procesado para la presentación de la información al usuario por lo cual se requiere verificar la conversión de esta lista de objetos a una lista de strings y estas sean ordenadas alfabéticamente.	
Responsable: Wilmer Barrera	**Fecha:** 20/11/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla RUTAS con datos - tener instalada la última versión de la app en el emulador	
Pasos de ejecución: - ejecutar la app desde el emulador - verificar el tipo de respuesta de conversión que se imprime en la consola	
Resultado esperado: cuando la conversión de objetos de tipo ruta se pasó a una lista de Strings, esta lista se debe imprimir en la consola, donde se presenta esta lista de manera ordenada alfabéticamente.	

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 19s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 04 de historia de usuario 06.

TAREA DE INGENIERÍA		
Id: TI_04	**Tipo de tarea:** desarrollo	**Sprint:** 4
Nombre de historia usuario: HU_06 Implementación de funcionalidad para consulta de horarios por transporte		
Nombre tarea: Crear funcionalidad para obtener lista de dependencias que pertenecen a una cooperativa de transporte.		
Fecha inicio: 21/11/17		**Fecha fin:**21/11/17
Descripción: Como desarrollador necesito crear la funcionalidad en la que se extraerán la lista de dependencias que pertenecen a una cooperativa de transporte desde la base de datos local		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_56	Verificar que se cargue lista de dependencias de una cooperativa de transporte
PA_57	Verificar existencia de dependencia a la que pertenece una ruta de una cooperativa de transporte
PA_58	Verificar cuando no existe dependencia a la que pertenece una ruta de una cooperativa de transporte
PA_59	Convertir lista de objetos de cooperativas de transporte a lista de strings
PA_60	Verificar cuando no existan dependencias en las que se encuentra registrada una ruta específica

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 56.

PRUEBA DE ACEPTACIÓN	
Id: PA_56	**Nombre:** Verificar que se cargue lista de dependencias de una cooperativa de transporte
Tarea de ingeniería: TI_04 Crear funcionalidad para obtener lista de dependencias que pertenecen a una cooperativa de transporte.	
Descripción: para poder mostrar la lista de dependencias al usuario se debe extraer esta información desde la base de datos.	
Responsable: Wilmer Barrera	**Fecha:** 21/11/17
Condición de ejecución:	

- Tener creada la base de datos - Tener creada la tabla DEPENDENCIAS - Tener instalada la última versión de la app en el emulador - Tener datos en la tabla dependencias
Pasos de ejecución: - Ejecutar la app desde el emulador - Verificar que se presente la información de la lista de dependencias que tienen una ruta en común
Resultado esperado: al consultar sobre todas las dependencias que tienen asignada una misma ruta, estas dependencias se deben presentar en modo consola.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 9s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 57.

PRUEBA DE ACEPTACIÓN	
Id: PA_57	**Nombre:** Verificar existencia de dependencia a la que pertenece una ruta de una cooperativa de transporte
Tarea de ingeniería: TI_04 Crear funcionalidad para obtener lista de dependencias que pertenecen a una cooperativa de transporte.	
Descripción: para poder mostrar la información de una agencia específica que tiene asignada una ut determinada debemos poder extraer la información de esta a partir de su id	
Responsable: Wilmer Barrera	**Fecha:** 21/11/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla DEPENDENCIA con datos - Tener instalada la última versión de la app	
Pasos de ejecución: - Ejecutar la app desde el emulador - Verificar el tipo de respuesta que se imprime en consola	
Resultado esperado: cuando se realice un proceso de consulta de una determinada dependencia este resultado se debe imprimir en modo consola, mostrando un número entero que corresponde al id de la dependencia.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 58.

PRUEBA DE ACEPTACIÓN	
Id: PA_58	**Nombre:** Verificar cuando no existe dependencia a la que pertenece una ruta de una cooperativa de transporte
Tarea de ingeniería: TI_04 Crear funcionalidad para obtener lista de dependencias que pertenecen a una cooperativa de transporte.	
Descripción: cuando se requiere encontrar una dependencia determinada y esta no se encuentre registrada, el sistema debe reaccionar ante este caso.	
Responsable: Wilmer Barrera	**Fecha:** 21/11/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla DEPENDENCIA sin datos - Tener instalada la última versión de la app en el emulador	
Pasos de ejecución: - Ejecutar aplicación desde el emulador - Verificar el tipo de respuesta que se obtiene en la consola	
Resultado esperado: cuando se realice una consulta sobre una dependencia determinada y esta no se encuentre registrada, el sistema mostrará un mensaje de advertencia donde nos dice que no se ha encontrado resultados	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 2s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 59.

PRUEBA DE ACEPTACIÓN	
Id: PA_59	**Nombre:** Convertir lista de objetos de cooperativas de transporte a lista de strings.
Tarea de ingeniería: TI_04 Crear funcionalidad para obtener lista de dependencias que pertenecen a una cooperativa de transporte.	
Descripción: cuando se ha extraído la lista de cooperativas desde la base de datos, esta se guarda en objetos, que deben ser adaptados o procesados para posteriormente presentarse en pantalla.	
Responsable: Wilmer Barrera	**Fecha:** 21/11/17
Condición de ejecución: - Tener creada la base de datos - Tener instalada la última versión de la app - Tener creada la tabla COOPERATIVA con datos.	

Pasos de ejecución: - Ejecutar la app desde el emulador - Verificar la respuesta de conversión de objetos cooperativa a lista de strings se impriman en modo consola.
Resultado esperado: cando se ha convertido con éxito una lista de objetos a una lista de cadenas de texto, esta cadena se mostrará impresa en modo consola.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** 23s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 60.

PRUEBA DE ACEPTACIÓN	
Id: PA_60	**Nombre:** Verificar cuando no existan dependencias en las que se encuentra registrada una ruta específica
Tarea de ingeniería: TI_04 Crear funcionalidad para obtener lista de dependencias que pertenecen a una cooperativa de transporte.	
Descripción: cuando se requiere buscar una dependencia y esta no existe entre la lista de objetos atraídos de la base de datos, y entre estos objetos no existe dicha dependencia el sistema debe mostrar un tipo de alerta	
Responsable: Wilmer Barrera	**Fecha:** 21/11/17
Condición de ejecución: - Tener creada la base de datos - Tener instalada la última versión de la base de datos - Tener desarrollada la funcionalidad de búsqueda de dependencia por id	
Pasos de ejecución: - Insertar el id de dependencia que no existe como parámetro de búsqueda - Ejecutar la aplicación desde el emulador - Verificar el tipo de respuesta presentado	
Resultado esperado: cuando se busque una dependencia dentro de una lista de objetos de dependencia y este no sea hallado, este proceso devolverá un resultado nulo el cual se mostrará en pantalla	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 05 de historia de usuario 06.

TAREA DE INGENIERÍA		
Id: TI_05	**Tipo de tarea:** desarrollo	**Sprint:** 4
Nombre de historia usuario: HU_06 Implementación de funcionalidad para consulta de horarios por transporte		
Nombre tarea: Crear funcionalidad para extracción de lista de horarios en base a una ruta de una agencia de cooperativa de transporte específica.		
Fecha inicio: 22/11/17		**Fecha fin:**22/11/17
Descripción: Como desarrollador necesito crear la funcionalidad en la que se extraerán la lista de horarios que pertenecen a una ruta específica desde la base de datos local		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_61	Verificar que se cargue lista de horarios de una ruta que pertenece a una agencia de cooperativa de transporte
PA_62	Verificar cuando la lista de horarios de una ruta que pertenece a una agencia de cooperativa de transporte está vacía

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 61.

PRUEBA DE ACEPTACIÓN	
Id: PA_61	**Nombre:** Verificar que se cargue lista de horarios de una ruta que pertenece a una agencia de cooperativa de transporte
Tarea de ingeniería: TI_05 Crear funcionalidad para extracción de lista de horarios en base a una ruta de una agencia de cooperativa de transporte específica.	
Descripción: al realizar consultas de horarios de transporte de una ruta específica esta debe devolver una matriz con todos aquellos horarios.	
Responsable: Wilmer Barrera	**Fecha:** 22/11/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla HORARIOS con datos - Tener instalada la última versión de la app móvil	
Pasos de ejecución: - Ingresar el id de la dependencia posee ruta como parámetro de búsqueda de horarios. - Ejecutar la app desde el emulador - Verificar el tipo de resultado que se muestra en consola	

Resultado esperado: cuando se lleva a cabo la extracción de horarios de una ruta este proceso devolverá una matriz no nula como resultado, si esta matriz no es nula mostrará un mensaje de confirmación de extracción.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 62.

PRUEBA DE ACEPTACIÓN	
Id: PA_62	**Nombre:** Verificar cuando la lista de horarios de una ruta que pertenece a una agencia de cooperativa de transporte está vacía
Tarea de ingeniería: TI_05 Crear funcionalidad para extracción de lista de horarios en base a una ruta de una agencia de cooperativa de transporte específica.	
Descripción: al realizar consultas de horarios de transporte de una ruta específica esta debe devolver una matriz nula dando a entender que no se han encontrado horarios.	
Responsable: Wilmer Barrera	**Fecha:** 22/11/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla HORARIOS sin datos - Tener instalada la última versión de la app móvil	
Pasos de ejecución: - Ingresar el id de la dependencia posee ruta como parámetro de búsqueda de horarios. - Ejecutar la app desde el emulador - Verificar el tipo de resultado que se muestra en consola	
Resultado esperado: cuando se lleva a cabo la extracción de horarios de una ruta este proceso devolverá una matriz nula como resultado, lo cual hará que se muestre un mensaje diciendo que no existen horarios.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 06 de historia de usuario 06.

TAREA DE INGENIERÍA		
Id: TI_06	**Tipo de tarea:** desarrollo	**Sprint:** 4
Nombre de historia usuario: HU_06 Implementación de funcionalidad para consulta de horarios por transporte		
Nombre tarea: Crear adaptador personalizado de listview para presentar lista de horarios de una ruta por transporte.		

Fecha inicio: 22/11/17	**Fecha fin:**22/11/17
Descripción: Con el fin de presentar la información de horarios se implementará un adaptador personalizado el cual presentará los horarios en un listview	
PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_63	Verificar que la información se presente de acuerdo al diseño requerido por el cliente.

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 63.

PRUEBA DE ACEPTACIÓN	
Id: PA_63	**Nombre:** Verificar que la información se presente de acuerdo al diseño requerido por el cliente.
Tarea de ingeniería: TI_06 Crear adaptador personalizado de listview para presentar lista de horarios de una ruta por transporte.	
Descripción: para mostrar la información acerca de horarios de rutas se debe seguir los bosquejos provistos por el cliente.	
Responsable: Wilmer Barrera	**Fecha:** 22/11/17
Condición de ejecución: - Tener los bosquejos de la app - Tener instalada la última versión de la app en el emulador - Tener la lista de colores en hexadecimal	
Pasos de ejecución: - Verificar que los colores sean los mismos de los bosquejos - Verificar que la distribución de objetos de interfaz esté distribuida de acuerdo a los bosquejos - Verificar tamaños de letras siguiendo bosquejos	
Resultado esperado: cuando se tiene implementada la interfaz debe ser visiblemente parecida al bosquejo provisto por el cliente.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Sprint 5

Tabla historia de usuario 07.

HISTORIA DE USUARIO		
Id: HU_07	**Nombre:** Implementación de interfaz favoritos	
Descripción: Como desarrollador necesito implementa una interfaz de usuario para presentar la información que contiene una ruta favorita que fue agregada por el usuario.		
Usuario: Wilmer Barrera	**Sprint:** 5	
Fecha inicio: 27/11/17	**Fecha fin:** 28/11/17	**Esfuerzo:** 10

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Diseño de interfaz para mostrar lista de rutas favoritas y sus detalles

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 07.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 5
Nombre de historia usuario: HU_07 Implementación de interfaz favoritos		
Nombre tarea: Diseño de interfaz para mostrar lista de rutas favoritas y sus detalles.		
Fecha inicio: 27/11/17		**Fecha fin:**28/11/17
Descripción: Como desarrollador necesito crear la interfaz donde se presentará la lista de rutas favoritas		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_64	Verificar que el diseño de la interfaz siga el bosquejo provisto por el cliente

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 64.

PRUEBA DE ACEPTACIÓN	
Id: PA_64	**Nombre:** Verificar que el diseño de la interfaz siga el bosquejo provisto por el cliente
Tarea de ingeniería: TI_01 Diseño de interfaz para mostrar lista de rutas favoritas y sus detalles	
Descripción: El diseño de la interfaz de usuario que muestre las rutas favoritas debe seguir los bosquejos provistos por el cliente.	
Responsable: Wilmer Barrera	**Fecha:** 28/11/17

Condición de ejecución: - Tener instalado el IDE de desarrollo Android Studio - Tener instalado un emulador para probar la app - Tener establecido un bosquejo en el cual se basará el diseño de la interfaz
Pasos de ejecución: - Verificar que la distribución de componentes de interfaz siga el bosquejo - Verificar que los colores de la interfaz sigan los bosquejos definidos por el cliente - Verificar que el texto siga el formato como se encuentra definido en el bosquejo.
Resultado esperado: La interfaz de usuario se basa completamente en el bosquejo provisto por el cliente

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla historia de usuario 08.

HISTORIA DE USUARIO		
Id: HU_08	**Nombre:** Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino	
Descripción: Como desarrollador necesito implementar la funcionalidad para la consulta de horarios en base a ruta específica sin importar a cuál cooperativa de transporte corresponda.		
Usuario: Wilmer Barrera	**Sprint:** 5	
Fecha inicio: 29/11/17	**Fecha fin:** 05/12/17	**Esfuerzo:** 25

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Creación de funcionalidad para extraer lista de ciudades origen desde la base de datos
TI_02	Creación de funcionalidad para extraer lista de ciudades destino basado en una ciudad origen
TI_03	Crear funcionalidad para formatear lista de objetos ruta a lista de strings
TI_04	Crear funcionalidad para extraer lista de horarios en base a una ruta, sin importar la cooperativa de transporte
TI_05	Crear adaptador personalizado para presentar lista de horarios en base a una ciudad origen y destino

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 08.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 5
Nombre de historia usuario: HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino		
Nombre tarea: Creación de funcionalidad para extraer lista de ciudades origen desde la base de datos.		
Fecha inicio: 29/11/17		**Fecha fin:**29/11/17
Descripción: Como desarrollador necesito crear la funcionalidad en la que se extraerán la lista de ciudades origen desde la base de datos local		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_65	Verificar que se cargue lista de ciudades origen desde la base de datos local	
PA_66	Verificar que la lista de ciudades origen esté vacía	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 65.

PRUEBA DE ACEPTACIÓN	
Id: PA_65	**Nombre:** Verificar que se cargue lista de ciudades origen desde la base de datos local
Tarea de ingeniería: TI_01 Creación de funcionalidad para extraer lista de ciudades origen desde la base de datos.	
Descripción: cuando se extrae la lista de ciudades origen de la tabla ruta se debe verificar que esta lista traiga datos y sean almacenados en una lista de strings	
Responsable: Wilmer Barrera	**Fecha:** 29/11/17
Condición de ejecución: - Tener creada la base de datos - Tener instalada la última versión de la app - Tener creada y con datos la tabla RUTA - Tener creada la conexión a la base de datos - Tener creada la funcionalidad para extraer ciudades origen de la tabla rutas	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar botón que muestre diálogo de lista de ciudades origen - Verificar respuestas imprimida en la consola	

Resultado esperado: al ejecutarse el proceso de cargar la lista de ciudades origen desde la tabla ruta, esta lista debe devolver un array de strings que no será nula, imprimiéndose de esta manera dicha lista en la consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 66.

PRUEBA DE ACEPTACIÓN	
Id: PA_66	**Nombre:** Verificar que la lista de ciudades origen esté vacía
Tarea de ingeniería: TI_01 Creación de funcionalidad para extraer lista de ciudades origen desde la base de datos.	
Descripción: cuando se extrae la lista de ciudades origen de la tabla ruta y esta no traiga datos, esta situación que presenta la app debe ser controlada.	
Responsable: Wilmer Barrera	**Fecha:** 29/11/17
Condición de ejecución: - Tener creada la base de datos - Tener instalada la última versión de la app - Tener creada y sin datos la tabla RUTA - Tener creada la conexión a la base de datos - Tener creada la funcionalidad para extraer ciudades origen de la tabla rutas	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar botón que muestre diálogo de lista de ciudades origen - Verificar respuestas imprimida en la consola	
Resultado esperado: al ejecutarse el proceso de cargar la lista de ciudades origen desde la tabla ruta, esta lista devuelve un valor nulo, que se imprimirá como un mensaje advirtiendo que no hay resultados, este se presentará en la consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 02 de historia de usuario 08.

TAREA DE INGENIERÍA		
Id: TI_02	**Tipo de tarea:** desarrollo	**Sprint:** 5
Nombre de historia usuario: HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino		
Nombre tarea: Creación de funcionalidad para extraer lista de ciudades destino basado en una ciudad origen.		
Fecha inicio: 30/11/17		**Fecha fin:**30/11/17
Descripción: Como desarrollador necesito crear la funcionalidad en la que se extraerán la lista de ciudades destino con base a una ciudad origen desde la base de datos local		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_67	Verificar que se cargue lista de ciudades destino con base en una ciudad origen desde la base de datos local	
PA_68	Verificar que la lista de ciudades destino esté vacía	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 67.

PRUEBA DE ACEPTACIÓN	
Id: PA_67	**Nombre:** Verificar que se cargue lista de ciudades destino con base en una ciudad origen desde la base de datos local
Tarea de ingeniería: TI_02 Creación de funcionalidad para extraer lista de ciudades destino basado en una ciudad origen.	
Descripción: cuando se extrae la lista de ciudades destino de la tabla ruta se debe verificar que esta lista traiga datos y sean almacenados en una lista de objetos ruta.	
Responsable: Wilmer Barrera	**Fecha:** 30/11/17
Condición de ejecución: - Tener creada la base de datos - Tener instalada la última versión de la app - Tener creada y con datos la tabla RUTA - Tener creada la conexión a la base de datos - Tener creada la funcionalidad para extraer ciudades destino de la tabla rutas	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar botón que muestre diálogo de lista de ciudades destino	

- Verificar respuestas imprimida en la consola
Resultado esperado: al ejecutarse el proceso de cargar la lista de ciudades destino desde la tabla ruta, esta lista debe devolver una lista de objetos ruta con un número entero de elementos, y este número de elementos será mostrado en la consola.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 68.

PRUEBA DE ACEPTACIÓN	
Id: PA_68	**Nombre:** Verificar que la lista de ciudades destino esté vacía
Tarea de ingeniería: TI_02 Creación de funcionalidad para extraer lista de ciudades destino basado en una ciudad origen.	
Descripción: cuando se extrae la lista de ciudades destino de la tabla ruta y esta no encuentre datos, el sistema debe reaccionar ante esta situación.	
Responsable: Wilmer Barrera	**Fecha:** 30/11/17
Condición de ejecución: - Tener creada la base de datos - Tener instalada la última versión de la app - Tener creada y con datos la tabla RUTA - Tener creada la conexión a la base de datos - Tener creada la funcionalidad para extraer ciudades destino de la tabla rutas	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar botón que muestre diálogo de lista de ciudades destino - Verificar respuestas imprimida en la consola	
Resultado esperado: al ejecutarse el proceso de cargar la lista de ciudades destino desde la tabla ruta y no se ha encontrado resultados este proceso devolverá un número entero de cero que corresponde al número de elementos existentes, y este número de elementos será mostrado en la consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 03 de historia de usuario 08.

TAREA DE INGENIERÍA		
Id: TI_03	**Tipo de tarea:** desarrollo	**Sprint:** 5
Nombre de historia usuario: HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino		
Nombre tarea: Crear funcionalidad para formatear lista de objetos ruta a lista de strings.		
Fecha inicio: 01/12/17		**Fecha fin:**01/12/17
Descripción: Con el objetivo de mostrar una lista de rutas a través de un cuadro de diálogo se formateará una lista de objetos de ruta a lista de strings.		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_69	Verificar que lista de objetos de rutas sea formateado correctamente a lista de strings
PA_70	Verificar formateo de campos de rutas empiecen con letra mayúscula

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 69.

PRUEBA DE ACEPTACIÓN	
Id: PA_69	**Nombre:** Verificar que lista de objetos de rutas sea formateado correctamente a lista de strings
Tarea de ingeniería: TI_03 Crear funcionalidad para formatear lista de objetos ruta a lista de strings	
Descripción: para poder mostrar en un dialogo la lista de ciudades destino primero necesita tener un formato de cadena donde solo se almacene el nombre las ciudades en una lista de strings	
Responsable: Wilmer Barrera	**Fecha:** 01/12/17
Condición de ejecución: - Tener instalada la última versión de la app - Tener implementada la funcionalidad para formatear objetos ruta a lista de strings - Tener la base de dato creada - Tener la tabla RUTA creada y con datos	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar el botón para mostrar ciudades destino - Verificar que el cuadro de dialogo se presenten los nombres de las ciudades destino	

Resultado esperado: cuando se presione el botón para mostrar las ciudades destino se debe mostrar en pantalla un cuadro de dialogo con la lista de ciudades destino.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 5s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 70.

PRUEBA DE ACEPTACIÓN	
Id: PA_70	**Nombre:** Verificar formateo de campos de rutas empiecen con letra mayúscula
Tarea de ingeniería: TI_03 Crear funcionalidad para formatear lista de objetos ruta a lista de strings	
Descripción: con el objetivo de cumplir con el formato de texto que se muestra en la app siguiendo los bosquejos provistos por el cliente se formateará que los nombres de las ciudades de las rutas empiecen con letra mayúscula.	
Responsable: Wilmer Barrera	**Fecha:** 01/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener creada la base de datos - Tener creada la tabla RUTA con datos - Tener implementada la funcionalidad para extracción de rutas desde la base de datos	
Pasos de ejecución: - Ingresar la ciudad origen como un parámetro de búsqueda de ciudades destino - Ejecutar la app desde el emulador - Presionar el botón para mandar a listar la lista de rutas en un cuadro de dialogo.	
Resultado esperado: cuando se presione el botón ara mostrar la lista de rutas esta debe presentar una lista ciudades destino las cuales empezarán con el formato de tener la primera letra mayúscula y las demás minúsculas.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 04 de historia de usuario 08.

TAREA DE INGENIERÍA		
Id: TI_04	**Tipo de tarea:** desarrollo	**Sprint:** 5
Nombre de historia usuario: HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino		
Nombre tarea: Crear funcionalidad para extraer lista de horarios en base a una ruta, sin importar la cooperativa de transporte.		

Fecha inicio: 04/12/17	**Fecha fin:**04/12/17
Descripción: Como desarrollador necesito crear la funcionalidad en la que se extraerán la lista de horarios de una ruta específica desde la base de datos local	

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_71	Verificar que se cargue lista de horarios de una ruta específica
PA_72	Verificar que lista de horarios de una ruta específica esté vacía
PA_73	Verificar cuando existe id de una ruta específica
PA_74	Verificar cuando no existe id de una ruta específica

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 71.

PRUEBA DE ACEPTACIÓN	
Id: PA_71	**Nombre:** Verificar que se cargue lista de horarios de una ruta específica
Tarea de ingeniería: TI_04 Crear funcionalidad para extraer lista de horarios en base a una ruta, sin importar la cooperativa de transporte.	
Descripción: cuando se extrae la lista de horarios de una ruta específica, se debe verificar que esta lista traiga datos y sean almacenados en una lista de objetos de horarios.	
Responsable: Wilmer Barrera	**Fecha:** 04/12/17
Condición de ejecución: - Tener creada la base de datos - Tener instalada la última versión de la app - Tener creada y con datos la tabla HORARIO - Tener creada la conexión a la base de datos - Tener creada la funcionalidad para extraer horarios en base a una ruta	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar botón que muestre la lista de horarios de una ruta específica. - Verificar respuestas imprimida en la consola	
Resultado esperado: al ejecutarse el proceso de cargar la lista de horarios de una ruta, esta lista debe devolver un array de objetos de horarios, esta lista posee un número entero de elementos que se presentará en modo consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 5s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 72.

PRUEBA DE ACEPTACIÓN	
Id: PA_72	**Nombre:** Verificar que lista de horarios de una ruta específica esté vacía
Tarea de ingeniería: TI_04 Crear funcionalidad para extraer lista de horarios en base a una ruta, sin importar la cooperativa de transporte.	
Descripción: cuando se extrae la lista de horarios de una ruta específica y no traiga datos ante esta situación el sistema debe responder de una manera.	
Responsable: Wilmer Barrera	**Fecha:** 04/12/17
Condición de ejecución: - Tener creada la base de datos - Tener instalada la última versión de la app - Tener creada y sin datos la tabla HORARIO - Tener creada la conexión a la base de datos - Tener creada la funcionalidad para extraer horarios en base a una ruta	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar botón que muestre la lista de horarios de una ruta específica. - Verificar respuestas imprimida en la consola	
Resultado esperado: al ejecutarse el proceso de cargar la lista de horarios de una ruta y en caso de no tener resultado devolverá un número entero de cero, este representa la cantidad de elementos de resultado y será presentado en la consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 73.

PRUEBA DE ACEPTACIÓN	
Id: PA_73	**Nombre:** Verificar cuando existe id de una ruta específica
Tarea de ingeniería: TI_04 Crear funcionalidad para extraer lista de horarios en base a una ruta, sin importar la cooperativa de transporte.	
Descripción: con el fin de obtener una ruta específica, se necesita realizar una búsqueda dentro de una lista de rutas ubicados en objetos.	
Responsable: Wilmer Barrera	**Fecha:** 04/12/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la conexión a la base de datos - Tener creada la tabla rutas con datos	

- Tener instalada la última versión de la app - Tener desarrollada la funcionalidad para buscar el id de una ruta en una lista de objetos ruta.
Pasos de ejecución: - Ingresar la ruta (ciudad origen y destino) a buscar como parámetro de búsqueda - Ejecutar la app en el emulador - Presionar el botón para buscar una ruta específica - Verificar el resultado devuelto e impreso en la consola
Resultado esperado: cuando se manda a buscar una ruta específica y se obtiene un resultado positivo, este resultado se mostrará en la consola, donde se presenta el id de la ruta buscada.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 74.

PRUEBA DE ACEPTACIÓN	
Id: PA_74	**Nombre:** Verificar cuando no existe id de una ruta específica
Tarea de ingeniería: TI_04 Crear funcionalidad para extraer lista de horarios en base a una ruta, sin importar la cooperativa de transporte.	
Descripción: cuando se busca una ruta específica, se necesita realizar una búsqueda dentro de una lista de rutas ubicados en objetos y en caso de no obtener resultado, la app debe reaccionar ante esta situación.	
Responsable: Wilmer Barrera	**Fecha:** 04/12/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la conexión a la base de datos - Tener creada la tabla rutas sin datos - Tener instalada la última versión de la app - Tener desarrollada la funcionalidad para buscar el id de una ruta en una lista de objetos ruta.	
Pasos de ejecución: - Ingresar la ruta (ciudad origen y destino) a buscar como parámetro de búsqueda - Ejecutar la app en el emulador - Presionar el botón para buscar una ruta específica - Verificar el resultado devuelto e impreso en la consola	

Resultado esperado: cuando se manda a buscar una ruta específica y se obtiene un resultado negativo, en este caso se mostrará en valor entero cero en la consola, ya que este número representa que no se encontró id de ruta alguna.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 05 de historia de usuario 08.

TAREA DE INGENIERÍA		
Id: TI_05	**Tipo de tarea:** desarrollo	**Sprint:** 5
Nombre de historia usuario: HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino		
Nombre tarea: Crear adaptador personalizado para presentar lista de horarios en base a una ciudad origen y destino.		
Fecha inicio: 05/12/17		**Fecha fin:**05/12/17
Descripción: A través de un adaptador personalizado que se implementará, este servirá para presentar una lista de horarios de una ruta específica.		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_75	Verificar que la lista de resultados se muestre de acuerdo al diseño requerido por el cliente.	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 75.

PRUEBA DE ACEPTACIÓN	
Id: PA_75	**Nombre:** Verificar que la lista de resultados se muestre de acuerdo al diseño requerido por el cliente.
Tarea de ingeniería: TI_05 Crear adaptador personalizado para presentar lista de horarios en base a una ciudad origen y destino.	
Descripción: con el fin de mostrar la información de los horarios de una ruta específica, esta se debe verificar que siga los diseños de los bosquejos provistos por el cliente, así como colores y formatos de texto	
Responsable: Wilmer Barrera	**Fecha:** 05/12/17
Condición de ejecución: - tener desarrollada la interfaz del adaptador para presentar horarios	
Pasos de ejecución:	

- verificar los colores de la interfaz - verificar la distribución de los elementos de interfaz - verificar el formato y tamaño de textos de la interfaz
Resultado esperado: al tener desarrollada la interfaz esta debe ser lo más parecida posible en modo visual al bosquejo provisto por el cliente.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla historia de usuario 09.

HISTORIA DE USUARIO		
Id: HU_09	**Nombre:** Implementación de funcionalidad para obtener la próxima salida en base a la hora actual	
Descripción: Como desarrollador necesito implementar la funcionalidad en la que se pueda obtener información sobre la próxima salida de una ruta específica, basado en la hora actual del dispositivo.		
Usuario: Wilmer Barrera	**Sprint:** 5	
Fecha inicio: 06/12/17	**Fecha fin:** 07/12/17	**Esfuerzo:** 10

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Implementar funcionalidad para extraer la fecha del sistema
TI_02	Implementar funcionalidad para ordenar horarios de rutas de transporte
TI_03	Implementar funcionalidad para extraer próxima salida

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 09.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 5
Nombre de historia usuario: HU_09 Implementación de funcionalidad para obtener la próxima salida en base a la hora actual		
Nombre tarea: Implementar funcionalidad para extraer la fecha del sistema.		
Fecha inicio: 06/12/17		**Fecha fin:**06/12/17
Descripción: Como desarrollador necesito crear la funcionalidad en la que se extraerán fecha actual del sistema, esto incluye tanto la fecha como la hora.		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	

PA_76	Verificar el formato de fecha es el correcto

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 76.

PRUEBA DE ACEPTACIÓN	
Id: PA_76	**Nombre:** Verificar el formato de fecha es el correcto
Tarea de ingeniería: TI_01 Implementar funcionalidad para extraer la fecha del sistema	
Descripción: cuando se extrae la fecha del sistema se necesita verificar que el formato sea correcto para poder llevar a cabo el cálculo de próxima salida.	
Responsable: Wilmer Barrera	**Fecha:** 06/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener configurada la hora del emulador correctamente	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar el botón para obtener la fecha actual del sistema - Verificar la forma en que se presenta el resultado de la fecha actual	
Resultado esperado: cuando se extrae la fecha del sistema se debe extraer en el formato siguiente, primero la hora, minutos, segundos seguido de esto la parte del día al que pertenece la hora AM o PM y seguido por el día, mes y año.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 02 de historia de usuario 09.

TAREA DE INGENIERÍA		
Id: TI_02	**Tipo de tarea:** desarrollo	**Sprint:** 5
Nombre de historia usuario: HU_09 Implementación de funcionalidad para obtener la próxima salida en base a la hora actual		
Nombre tarea: Implementar funcionalidad para ordenar horarios de rutas de transporte.		
Fecha inicio: 06/12/17		**Fecha fin:**07/12/17
Descripción: Con el fin de presentar la lista de horarios ordenados ascendentemente, la lista de horarios será ordenado para que el usuario pueda encontrar un horario adecuado de manera fácil		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	

PA_77	Verificar adaptación de lista de objetos de horarios y rutas a lista adaptada de listview para presentación en interfaz
PA_78	Verificar que lista de objetos se ordenen ascendentemente de acuerdo al horario
PA_79	Verificar que la extracción de la hora actual sea correcta

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 77.

PRUEBA DE ACEPTACIÓN	
Id: PA_77	**Nombre:** Verificar adaptación de lista de objetos de horarios y rutas a lista adaptada de listview para presentación en interfaz
Tarea de ingeniería: TI_02 Implementar funcionalidad para ordenar horarios de rutas de transporte	
Descripción: para la presentación de la lista de rutas de una o varias cooperativas de transporte estas deben ser formateadas de manera que se pueda presentar al usuario.	
Responsable: Wilmer Barrera	**Fecha:** 07/12/17
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla RUTA con dato - Tener creada la conexión a la base de datos - Tener instalada la última versión de la app en el emulador	
Pasos de ejecución: - Ejecutar la aplicación desde el emulador - Ver el resultado que se muestra en la consola	
Resultado esperado: cuando se realiza el proceso de adaptar una lista de rutas con sus horarios estos deben devolver una lista adaptada de rutas, la misma que muestra un número entero que muestra la cantidad de elementos que fueron adaptados, en tal caso en número es mayor que cero	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 78.

PRUEBA DE ACEPTACIÓN	
Id: PA_78	**Nombre:** Verificar que lista de objetos se ordenen ascendentemente de acuerdo al horario
Tarea de ingeniería: TI_02 Implementar funcionalidad para ordenar horarios de rutas de transporte	
Descripción: para mejor la presentación al usuario y mejor la rapidez de búsqueda de horarios estos deben ser ordenados ascendentemente.	
Responsable: Wilmer Barrera	**Fecha:** 07/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener creada la base de datos - Tener creada la tabla RUTA y HORARIO con datos - Tener creada la conexión a la base de datos.	
Pasos de ejecución: - Ingresar los parámetros de ruta para cargar lista de objetos ruta con horarios - Ejecutar la app desde el emulador - Presionar el botón para cargar la lista de horarios de una ruta	
Resultado esperado: cuando se presione el botón para presentar la lista de horarios de una ruta, esta lista debe presentarse en un listview ordenado ascendentemente.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 9s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 79.

PRUEBA DE ACEPTACIÓN	
Id: PA_79	**Nombre:** Verificar que la extracción de la hora actual sea correcta
Tarea de ingeniería: TI_02 Implementar funcionalidad para ordenar horarios de rutas de transporte	
Descripción: cuando se extraiga la hora del sistema para realizar búsquedas de la próxima salida esta deberá tener un formato correcto para poder acceder a sus datos de manera unificada.	
Responsable: Wilmer Barrera	**Fecha:** 07/12/17
Condición de ejecución: - Tener instalada la última versión de la app - Tener configurada correctamente la fecha del sistema	
Pasos de ejecución: - Ejecutar la aplicación desde el emulador	

- Presionar el emulador para obtener la hora actual del sistema - Verificar el formato de la respuesta obtenida del sistema
Resultado esperado: cuando se extrae la hora actual del sistema este debe tener el siguiente formato, comienza por dar la hora, minutos y segundos de la hora actual, seguido de esto debe estar la parte del día que corresponde la hora AM o PM.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 03 de historia de usuario 09.

TAREA DE INGENIERÍA		
Id: TI_03	**Tipo de tarea:** desarrollo	**Sprint:** 5
Nombre de historia usuario: HU_09 Implementación de funcionalidad para obtener la próxima salida en base a la hora actual		
Nombre tarea: Implementar funcionalidad para extraer próxima salida.		
Fecha inicio: 07/12/17		**Fecha fin:**07/12/17
Descripción: Con el fin de obtener la próxima salida en horario de una ruta se implementará la funcionalidad en la cual muestre la siguiente salida basada en la hora actual del dispositivo.		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_80	Verificar que la hora del día sea PM
PA_81	Verificar que la hora del día sea AM
PA_82	Verificar la extracción del día de la semana actual sea correcto
PA_83	Verificar conversión de día numérico a texto

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 80.

PRUEBA DE ACEPTACIÓN	
Id: PA_80	**Nombre:** Verificar que la hora del día sea PM
Tarea de ingeniería: TI_03 Implementar funcionalidad para extraer próxima salida	
Descripción: para poder verificar la parte del día a que corresponde una hora se deberá extraer el dato unificado que correspondiente.	
Responsable: Wilmer Barrera	**Fecha:** 07/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener configurada correctamente la hora del sistema	

Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar el botón para obtener la hora del sistema - Verificar el resultado que se muestra en la consola
Resultado esperado: cuando se manda a obtener la hora actual del sistema del emulador se extraerá la hora actual del sistema y se extraerá la información para verificar que, si es PM, pasado el meridiano.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** 2s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 81.

PRUEBA DE ACEPTACIÓN	
Id: PA_81	**Nombre:** Verificar que la hora del día sea AM
Tarea de ingeniería: TI_03 Implementar funcionalidad para extraer próxima salida	
Descripción: para poder verificar la parte del día a que corresponde una hora se deberá extraer el dato unificado que correspondiente.	
Responsable: Wilmer Barrera	**Fecha:** 07/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener configurada correctamente la hora del sistema	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar el botón para obtener la hora del sistema - Verificar el resultado que se muestra en la consola	
Resultado esperado: cuando se manda a obtener la hora actual del sistema del emulador se extraerá la hora actual del sistema y se extraerá la información para verificar que, si es AM, antes del meridiano.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 82.

PRUEBA DE ACEPTACIÓN	
Id: PA_82	**Nombre:** Verificar la extracción del día de la semana actual sea correcto
Tarea de ingeniería: TI_03 Implementar funcionalidad para extraer próxima salida	
Descripción: para poder verificar los días laborables de la semana se necesita saber qué día de la semana es el actual extraído del sistema	

Responsable: Wilmer Barrera	**Fecha:** 07/12/17
Condición de ejecución: - Tener instalada la última versión de la app - Tener configurada correctamente la fecha del sistema - Tener implementada la funcionalidad para extraer el día del sistema	
Pasos de ejecución: - Ejecutar la para desde el emulador - Presionar el botón para extraer la hora del sistema - Verificar la respuesta que muestra en la consola	
Resultado esperado: verificar que cuando se presente la información de respuesta en consola se presentará un número entero correspondiente al día de la semana, comenzando por lunes 1 y domingo como 7.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 83.

PRUEBA DE ACEPTACIÓN	
Id: PA_83	**Nombre:** Verificar conversión de día numérico a texto
Tarea de ingeniería: TI_03 Implementar funcionalidad para extraer próxima salida	
Descripción: para poder presentar los días laborables es necesario pasar el día numérico a día en texto, para que de esta manera el usuario pueda familiarizar mejor la información.	
Responsable: Wilmer Barrera	**Fecha:** 07/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener configurada correctamente la fecha del sistema.	
Pasos de ejecución: - Ejecutar la aplicación desde el emulador - Presionar el botón para obtener la hora actual del sistema. - Verificar el tipo de respuesta que se muestra en la consola	
Resultado esperado: cuando se presiones el botón para obtener el día actual de la semana, este día será extraído numéricamente, el mismo que luego será transformado a texto para mostrar al usuario.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Sprint 6

Tabla historia de usuario 10.

HISTORIA DE USUARIO		
Id: HU_10	**Nombre:** Creación de servicios web en PHP para sincronización con app móvil	
Descripción: Como desarrollador necesito implementar los servicios web php, los mismos que servirán para sincronizar las bases de datos de la aplicación, la base de datos local con la base de datos del servicio de hosting.		
Usuario: Wilmer Barrera	**Sprint:** 6	
Fecha inicio: 11/12/17	**Fecha fin:** 15/12/17	**Esfuerzo:** 25

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Implementación de funcionalidad para conectarse con la base de datos en hosting
TI_02	Implementación de funcionalidad para realizar transacciones en la base de datos de hosting
TI_03	Implementación de servicios web php para sincronizar base de datos local con la del servicio de hosting

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 10.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 6
Nombre de historia usuario: HU_10 Creación de servicios web en PHP para sincronización con app móvil		
Nombre tarea: Implementación de funcionalidad para conectarse con la base de datos en hosting.		
Fecha inicio: 11/12/17		**Fecha fin:**11/12/17
Descripción: Como desarrollador necesito crear un archivo de conexión directo a la base de datos, el cual servirá para realizar distintas transacciones a la misma.		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_84	Verificar que se conecte a la base de datos con las credenciales definidas
PA_85	Verificar que dicha funcionalidad permita realizar transacciones a la base de datos

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 84.

PRUEBA DE ACEPTACIÓN	
Id: PA_84	**Nombre:** Verificar que se conecte a la base de datos con las credenciales definidas
Tarea de ingeniería: TI_01 Implementación de funcionalidad para conectarse con la base de datos en hosting.	
Descripción: Para el consumo de servicios web es necesario enviar credenciales de autentificación con la cual se da acceso a los servicios web	
Responsable: Wilmer Barrera	**Fecha:** 11/12/17
Condición de ejecución: - Implementada la base de datos. - Tener definida las credenciales válidas para acceder a la base de datos	
Pasos de ejecución: - Cargar el archivo de conexión con el servicio de hosting - Abrir el archivo de conexión de servicio de hosting - Ejecutar clase de prueba de conexión a la base de datos.	
Resultado esperado: Las clases de conexión a la base de datos se conectan correctamente.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 85.

PRUEBA DE ACEPTACIÓN	
Id: PA_85	**Nombre:** Verificar que dicha funcionalidad permita realizar transacciones a la base de datos
Tarea de ingeniería: TI_01 Implementación de funcionalidad para conectarse con la base de datos en hosting.	
Descripción: Los scripts para consulta modificación inserción y eliminación debe funcionar correctamente.	
Responsable: Wilmer Barrera	**Fecha:** 11/12/17
Condición de ejecución: - Tener implementada la base de datos. - Tener implementado los archivos de conexión a la base de datos.	
Pasos de ejecución: - Verificar la implementación de scripts para inserción en cada tabla. - Verifica la implementación de scripts para eliminación en cada tabla. - Verificar la implementación de scripts para modificación cada tabla.	

- Verificar la implementación de scripts para consulta en cada tabla.
Resultado esperado: Los scripts inserción, eliminación, modificación y consulta han sido desarrollados en los archivos de transacciones de la base de datos.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 02 de historia de usuario 10.

TAREA DE INGENIERÍA		
Id: TI_02	**Tipo de tarea:** desarrollo	**Sprint:** 6
Nombre de historia usuario: HU_10 Creación de servicios web en PHP para sincronización con app móvil		
Nombre tarea: Implementación de funcionalidad para realizar transacciones en la base de datos de hosting.		
Fecha inicio: 12/12/17		**Fecha fin:**13/12/17
Descripción: Con el fin de realizar las distintas transacciones dentro de la base de datos se implementará las funcionalidades necesarias para poder realizar consultas a la base de datos.		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_86	Verificar que cada una de las transacciones funcionen de manera correcta según lo esperado
PA_87	Verificar que dicha funcionalidad pueda ser reusable desde otras clases

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 86.

PRUEBA DE ACEPTACIÓN	
Id: PA_86	**Nombre:** Verificar que cada una de las transacciones funcionen de manera correcta según lo esperado
Tarea de ingeniería: TI_02 Implementación de funcionalidad para realizar transacciones en la base de datos de hosting.	
Descripción: Las consultas y transacciones debe funcionar correctamente en cada una de las clases en las que se utiliza	
Responsable: Wilmer Barrera	**Fecha:** 13/12/17
Condición de ejecución: - Tener desarrollada la base de datos. - Tener creados los archivos de conexión a la base de datos.	

- Tener creados los archivos de transacciones a la base de datos.
Pasos de ejecución: - Llamar desde una clase externa al archivo de transacciones de base de datos y hacer una consulta en la misma. - Verificar que dicha consulta devuelva los datos correctos
Resultado esperado: Cada una de las transacciones devuelven los datos correctos.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 87.

PRUEBA DE ACEPTACIÓN	
Id: PA_87	**Nombre:** Verificar que dicha funcionalidad pueda ser reusable desde otras clases
Tarea de ingeniería: TI_02 Implementación de funcionalidad para realizar transacciones en la base de datos de hosting.	
Descripción: Se debe permitir la herencia de clases, para poder realizar transacciones desde cualquier clase externa.	
Responsable: Wilmer Barrera	**Fecha:** 13/12/17
Condición de ejecución: - Tener creada la base de datos. - Tener creados los archivos de conexión a la base de datos. - Tener creados los archivos de transacciones de la base de datos.	
Pasos de ejecución: - Crear una clase de prueba. - Heredar la clase de transacciones en la clase de prueba. - Verificar que se pueda realizar transacciones de la clase de prueba	
Resultado esperado: La herencia de la clase de transacciones es correcta, permitiendo reusar estas clases.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 03 de historia de usuario 10.

TAREA DE INGENIERÍA		
Id: TI_03	**Tipo de tarea:** desarrollo	**Sprint:** 6
Nombre de historia usuario: HU_10 Creación de servicios web en PHP para sincronización con app móvil		

Nombre tarea: Implementación de servicios web php para sincronizar base de datos local con la del servicio de hosting.	
Fecha inicio: 14/12/17	**Fecha fin:**15/12/17
Descripción: Con el fin de realizar transacciones a la base de datos del hosting desde la aplicación móvil se implementará los servicios web sobre php, el cual servirá como intermediario entre la aplicación móvil y la base de datos del servicio de hosting.	

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_88	Verificar que los servicios web hagan un control de autentificación para controlar el acceso a los servicios web del hosting
PA_89	Verificar que se tenga acceso a clases con las que se realizará dichas transacciones en la base de datos

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 88.

PRUEBA DE ACEPTACIÓN	
Id: PA_88	**Nombre:** Verificar que los servicios web hagan un control de autentificación para controlar el acceso a los servicios web del hosting
Tarea de ingeniería: TI_03 Implementación de servicios web php para sincronizar base de datos local con la del servicio de hosting.	
Descripción: Los servicios web se encuentran protegidos para que solo la aplicación móvil puede acceder a estos.	
Responsable: Wilmer Barrera	**Fecha:** 15/12/17
Condición de ejecución: - Tener creado los archivos de otra canción. - Tener creada la conexión desde la aplicación móvil al servicio web. - Tener definida las credenciales para acceder al servicio web.	
Pasos de ejecución: - Verificar el método de respuestas del servicio web. - Comparar credenciales de la aplicación móvil con las del servicio web. - Verificar extracción de parámetros en dicha petición.	
Resultado esperado: Limpio y la extracción de credenciales es correcto. Esto permite que se accede al consumo de los servicios web	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 89.

PRUEBA DE ACEPTACIÓN	
Id: PA_89	**Nombre:** Verificar que se tenga acceso a clases con las que se realizará dichas transacciones en la base de datos
Tarea de ingeniería: TI_03 Implementación de servicios web php para sincronizar base de datos local con la del servicio de hosting.	
Descripción: Desde la aplicación móvil se debe permitir realizar peticiones de transacciones a la base de datos desde los servicios web.	
Responsable: Wilmer Barrera	**Fecha:** 15/12/17
Condición de ejecución: - Llamar a implementado una conexión a los servicios web desde la aplicación móvil. - Tener creados los servicios web. - Tener creado los métodos de autenticación.	
Pasos de ejecución: - Instalar la última versión de la aplicación. - Realizar una petición de consulta a la base de datos. - Verificar respuestas del servicio web. - Verificar sí se accedió o no a los servicios web.	
Resultado esperado: Desde la aplicación móvil se puede acceder a los servicios web del servicio de hosting.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla historia de usuario 11.

HISTORIA DE USUARIO		
Id: HU_11	**Nombre:** Implementación de funcionalidad para mostrar horarios por días	
Descripción: Como desarrollador necesito implementar la funcionalidad que permita mostrar una lista de horarios con los días laborables a los que corresponde, esto en forma de texto tanto en inglés como en español.		
Usuario: Wilmer Barrera	**Sprint:** 6	
Fecha inicio: 18/12/17	**Fecha fin:** 21/12/17	**Esfuerzo:** 20

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Convertir días numéricos a días en forma de texto inglés y español

TI_02	Creación de funcionalidad para agrupación de días laborables de una agencia de una cooperativa de transporte

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 11.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 6
Nombre de historia usuario: HU_11 Implementación de funcionalidad para mostrar horarios por días		
Nombre tarea: Convertir días numéricos a días en forma de texto inglés y español.		
Fecha inicio: 18/12/17		**Fecha fin:**19/12/17
Descripción: Como desarrollador necesito implementar una funcionalidad en la cual se pueda convertir días numéricos a días en forma de texto, en idioma español e inglés		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_90	Verificar que un día numérico sea convertido a abreviación de día en texto tanto en español como en inglés	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 90.

PRUEBA DE ACEPTACIÓN	
Id: PA_90	**Nombre:** Verificar que un día numérico sea convertido a abreviación de día en texto tanto en español como en inglés
Tarea de ingeniería: TI_01 Convertir días numéricos a días en forma de texto inglés y español	
Descripción: para poder presentar la información de los días laborables que corresponden a los horarios de una ruta, esta debe presentarse en manera de texto abreviado tanto en inglés como en español	
Responsable: Wilmer Barrera	**Fecha:** 19/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener configurada correctamente la fecha del sistema	
Pasos de ejecución: - Ingresar el día para obtener la abreviación del día, como parámetro de conversión - Ejecutar la app desde el emulador - Presionar el botón para obtener la abreviación de día numérico	

- Verificar el resultado presentado en la consola
Resultado esperado: cuando se presenta la información en la consola el día numérico debe ser presentado en abreviación mostrando las 3 primeras letras del día sea en inglés o español, ejemplo 1: lun, 1: mon, …, 7: dom, 7: sun.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 02 de historia de usuario 11.

TAREA DE INGENIERÍA		
Id: TI_02	**Tipo de tarea:** desarrollo	**Sprint:** 6
Nombre de historia usuario: HU_11 Implementación de funcionalidad para mostrar horarios por días		
Nombre tarea: Creación de funcionalidad para agrupación de días laborables de una agencia de una cooperativa de transporte.		
Fecha inicio: 20/12/17		**Fecha fin:**21/12/17
Descripción: Como desarrollador necesito crear la funcionalidad en la cual me permita presentar el conjunto de días laborables de una cooperativa de transporte, estos días se presentarán de forma abreviada, mostrando solamente las 3 primeras letras de cada día.		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_91	Verificar la presentación de días laborables concatenados a partir de un conjunto de días numéricos.	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 91.

PRUEBA DE ACEPTACIÓN	
Id: PA_91	**Nombre:** Verificar la presentación de días laborables concatenados a partir de un conjunto de días numéricos.
Tarea de ingeniería: TI_02 Creación de funcionalidad para agrupación de días laborables de una agencia de una cooperativa de transporte.	
Descripción: para poder presentar el conjunto de días laborables de los horarios que cubren una ruta, estos deben estar unidos para presentación al usuario	
Responsable: Wilmer Barrera	**Fecha:** 21/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador	

- Tener configurada correctamente la fecha del sistema
Pasos de ejecución: - Ingresar el número que corresponde a los días laborables del horario de la ruta como parámetro. - Ejecutar la app desde el emulador - Presionar el botón para obtener la abreviación de día numérico - Verificar el resultado presentado en la consola
Resultado esperado: cuando se va a mostrar el conjunto de días laborables se debe mandar el número correspondiente a los días laborables ejemplo: 12567 corresponden a una concatenación de días abreviados conjuntamente ya sea en español o inglés y separados por el símbolo pai (|), ejemplo: Lun|Mar|Vié|Sáb|Dom o Mon|Tue|Fri|Sat|Sun.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 6s

Realizado por: Wilmer B. 2018.

Sprint 7

Tabla historia de usuario 12.

HISTORIA DE USUARIO		
Id: HU_12	**Nombre:** Implementación de funcionalidad para obtener coordenadas geográficas en Android	
Descripción: Como desarrollador necesito implementar la funcionalidad que me permita obtener coordenadas geográficas con el dispositivo celular, las coordenadas tomadas en cuenta serán la latitud y longitud.		
Usuario: Wilmer Barrera	**Sprint:** 7	
Fecha inicio: 26/12/17	**Fecha fin:** 28/12/17	**Esfuerzo:** 15

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Implementación de funcionalidad para usar el servicio de localización a través del dispositivo GPS
TI_02	Implementación de funcionalidad para usar el servicio de localización a través de la red celular a la que se encuentra conectado

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 12.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 7
Nombre de historia usuario: HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android		
Nombre tarea: Implementación de funcionalidad para usar el servicio de localización a través del dispositivo GPS.		
Fecha inicio: 26/12/17		**Fecha fin:**27/12/17
Descripción: Como desarrollador necesito crear la funcionalidad en la cual me permita usar el servicio de localización a través del dispositivo GPS del celular.		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_92	Verificar activación de servicio para localización
PA_93	Verificar que el proveedor de GPS esté activo
PA_94	Verificar que el proveedor de GPS esté inactivo
PA_95	Verificar omisión de altitud en localización
PA_96	Verificar consumo de energía bajo activo
PA_97	Verificar si existe proveedor de localización activo
PA_98	Verificar cuando no existe proveedor de localización activo
PA_99	Verificar desactivación de servicio de localización
PA_100	Verificar que el mejor proveedor de localización no sea el de red
PA_101	Verificar que el proveedor de localización esté en modo pasivo
PA_102	Verificar que el mejor proveedor de localización sea el GPS

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 92.

PRUEBA DE ACEPTACIÓN	
Id: PA_92	**Nombre:** Verificar activación de servicio para localización
Tarea de ingeniería: TI_01 Implementación de funcionalidad para usar el servicio de localización a través del dispositivo GPS.	
Descripción: para poder utilizar el servicio de localización ya sea a través de la red o el dispositivo GPS, antes se debe verificar que se ha activado este servicio.	
Responsable: Wilmer Barrera	**Fecha:** 27/12/17
Condición de ejecución: - Tener declarado permiso de localización en la app	

- Tener instalada la última versión de la app en el emulador - Tener desarrollada la funcionalidad para activar el servicio de localización
Pasos de ejecución: - Ejecutar la aplicación desde el emulador - Presionar botón para activar servicio de localización - Verificar el tipo de respuesta que nos muestra en consola
Resultado esperado: cuando se presione el botón para activar el servicio de localización, este proceso devolverá un resultado booleano verdadero que será interpretado como servicio de localización activado, esto a través de un mensaje.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 93.

PRUEBA DE ACEPTACIÓN	
Id: PA_93	**Nombre:** Verificar que el proveedor de GPS esté activo
Tarea de ingeniería: TI_01 Implementación de funcionalidad para usar el servicio de localización a través del dispositivo GPS.	
Descripción: para poder realizar una localización con el dispositivo GPS se debe controlar antes que debe estar activo.	
Responsable: Wilmer Barrera	**Fecha:** 27/12/17
Condición de ejecución: - Tener declarado permiso de localización en la app - Tener instalada la última versión de la app en el emulador - Activar la localización GPS en el emulador	
Pasos de ejecución: - Ejecutar la aplicación desde el emulador - Presionar botón para activar servicio de localización a través de GPS - Verificar el tipo de respuesta que nos muestra en consola	
Resultado esperado: cuando se presione el botón para verificar el servicio de localización a través del GPS este proceso devolverá un resultado booleano verdadero que será interpretado como servicio de localización por GPS activado, esto a través de un mensaje en consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 94.

PRUEBA DE ACEPTACIÓN	
Id: PA_94	**Nombre:** Verificar que el proveedor de GPS esté inactivo
Tarea de ingeniería: TI_01 Implementación de funcionalidad para usar el servicio de localización a través del dispositivo GPS.	
Descripción: para poder realizar una localización con el dispositivo GPS se debe controlar si el dispositivo GPS se encuentra inactivo.	
Responsable: Wilmer Barrera	**Fecha:** 27/12/17
Condición de ejecución: - Tener declarado permiso de localización en la app - Tener instalada la última versión de la app en el emulador - Desactivar la localización GPS en el emulador	
Pasos de ejecución: - Ejecutar la aplicación desde el emulador - Presionar botón para activar servicio de localización a través de GPS - Verificar el tipo de respuesta que nos muestra en consola	
Resultado esperado: cuando se presione el botón para verificar el servicio de localización a través del GPS este proceso devolverá un resultado booleano falso que será interpretado como servicio de localización por GPS inactivo, esto a través de un mensaje en consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 95.

PRUEBA DE ACEPTACIÓN	
Id: PA_95	**Nombre:** Verificar omisión de altitud en localización
Tarea de ingeniería: TI_01 Implementación de funcionalidad para usar el servicio de localización a través del dispositivo GPS.	
Descripción: para poder tener una respuesta más rápida de localización se debe omitir los datos que no se utilizarán, este es el caso de la altitud que brinda el servicio de localización GPS, este dato se debe omitir ya que no se usará	
Responsable: Wilmer Barrera	**Fecha:** 27/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador. - Tener desarrollada la funcionalidad de localización a través de GPS - Activar el dispositivo GPS del emulador.	
Pasos de ejecución:	

- Ejecutar la app desde el emulador - Presionar botón para verificar si se encuentra activa la localización por GPS - Verificar el tipo de respuesta que se muestra en la consola.
Resultado esperado: cuando se presione el botón para verificar la localización por GPS y se omita el dato de latitud este devolverá un valor booleano true que quiere decir que si se omitió dicho valor.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 96.

PRUEBA DE ACEPTACIÓN	
Id: PA_96	**Nombre:** Verificar consumo de energía bajo activo
Tarea de ingeniería: TI_01 Implementación de funcionalidad para usar el servicio de localización a través del dispositivo GPS.	
Descripción: para un consumo óptimo de la energía del dispositivo celular se debe programar un consumo bajo de energía, para que solo cuando la aplicación esté activa se lleve a cabo el proceso de localización.	
Responsable: Wilmer Barrera	**Fecha:** 27/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desarrollada la funcionalidad para localizar a través del GPS - Tener el dispositivo GPS del emulador activado.	
Pasos de ejecución: - Ejecutar la aplicación desde el emulador - Presionar el botón para la localización por GPS - Verificar el resultado que muestra la consola	
Resultado esperado: al presionar el botón activar localización por GPS, este proceso arroja un resultado con valor booleano true, en caso de haberse programado correctamente el consumo de energía bajo.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 97.

PRUEBA DE ACEPTACIÓN	
Id: PA_97	**Nombre:** Verificar si existe proveedor de localización activo
Tarea de ingeniería: TI_01 Implementación de funcionalidad para usar el servicio de localización a través del dispositivo GPS.	
Descripción: para poder llevar a cabo la localización ya sea por GPS o red hace falta verificar si, uno de estos métodos de localización está activo.	
Responsable: Wilmer Barrera	**Fecha:** 27/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desarrollada la funcionalidad para activar servicio de localización por GPS o red - Tener desarrollada la funcionalidad para verificar que existe un proveedor de localización activo - Tener activado el dispositivo GPS - Tener activado el consumo de datos móviles	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar el botón para verificar la localización - Verificar el tipo de resultado que se muestra en pantalla.	
Resultado esperado: cuando se lleve el proceso de verificar si existe algún proveedor de localización activo este proceso devolverá un valor booleano true, que mostrará un mensaje confirmando que, si existe un proveedor de localización activo, sea este el proveedor de red o GPS	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 98.

PRUEBA DE ACEPTACIÓN	
Id: PA_98	**Nombre:** Verificar cuando no existe proveedor de localización activo
Tarea de ingeniería: TI_01 Implementación de funcionalidad para usar el servicio de localización a través del dispositivo GPS.	
Descripción: para poder llevar a cabo la localización ya sea por GPS o red hace falta verificar cuan no existe un proveedor localización está activo.	
Responsable: Wilmer Barrera	**Fecha:** 27/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador	

- Tener desarrollada la funcionalidad para activar servicio de localización por GPS o red - Tener desarrollada la funcionalidad para verificar que existe un proveedor de localización activo - Tener desactivado el dispositivo GPS - Tener desactivado el consumo de datos móviles
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar el botón para verificar la localización - Verificar el tipo de resultado que se muestra en pantalla.
Resultado esperado: cuando se lleve el proceso de verificarla existencia de algún proveedor de localización activo este proceso devolverá un valor booleano false, que mostrará un mensaje advirtiendo que, no existe un proveedor de localización activo, sea este el proveedor de red o GPS

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 2s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 99.

PRUEBA DE ACEPTACIÓN	
Id: PA_99	**Nombre:** Verificar desactivación de servicio de localización
Tarea de ingeniería: TI_01 Implementación de funcionalidad para usar el servicio de localización a través del dispositivo GPS.	
Descripción: uno de los procesos con el fin de optimizar los recursos de la app móvil, el proceso es desactivar el servicio de localización para dejar de consumir recursos como energía del dispositivo	
Responsable: Wilmer Barrera	**Fecha:** 27/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador. - Tener desarrollada la funcionalidad para activar y desactivar el servicio de localización - Tener activado el dispositivo GPS o los datos móviles del emulador	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar el botón para activar y desactivar el servicio de localización - Verificar la respuesta que se muestra en la consola	
Resultado esperado: cuando se desactive el servicio de localización este proceso devolverá un valor booleano true, el cual se imprime como mensaje de confirmación que dice que el servicio de localización fue desactivado.	

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 100.

PRUEBA DE ACEPTACIÓN	
Id: PA_100	**Nombre:** Verificar que el mejor proveedor de localización no sea el de red
Tarea de ingeniería: TI_01 Implementación de funcionalidad para usar el servicio de localización a través del dispositivo GPS.	
Descripción: cuando se necesite realizar una localización ya sea de red o por GPS y se necesita tener una mejor precisión de la localización se debe controlar que no sea el proveedor de red el que se encuentra activo	
Responsable: Wilmer Barrera	**Fecha:** 27/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desarrollada la app para activar servicio de localización - Tener desarrollada la funcionalidad para verificar que no sea el proveedor de red el que se encuentra activo	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar el botón para activar el servicio de localización - Verificar la respuesta que se muestra en consola.	
Resultado esperado: cuando se manda a verificar que el proveedor de localización no sea GPS ni modo pasivo, este proceso debe dar como resultado una cadena de texto diferente a "network" que se mostrará en pantalla.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 101.

PRUEBA DE ACEPTACIÓN	
Id: PA_101	**Nombre:** Verificar que el proveedor de localización esté en modo pasivo
Tarea de ingeniería: TI_01 Implementación de funcionalidad para usar el servicio de localización a través del dispositivo GPS.	
Descripción: para poder verificar que el proveedor de localización está en modo pasivo se debe verificar para no iniciar un nuevo servicio de localización.	
Responsable: Wilmer Barrera	**Fecha:** 27/12/17
Condición de ejecución:	

- Tener instalada la última versión de la app en el emulador - Tener desarrollada la funcionalidad para activar servicio de localización - Tener activo el dispositivo GPS o datos móviles
Pasos de ejecución: - Ejecutar la aplicación móvil desde el emulador - Presionar botón para activar servicio de localización - Verificar el tipo de respuesta que se presenta en consola.
Resultado esperado: al verificar si el mejor proveedor de internet está en modo pasivo, la respuesta del mejor proveedor deberá mostrar una cadena de texto igual a" passive" el cual se mostrará en la consola de la app.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 102.

PRUEBA DE ACEPTACIÓN	
Id: PA_102	**Nombre:** Verificar que el mejor proveedor de localización sea el GPS
Tarea de ingeniería: TI_01 Implementación de funcionalidad para usar el servicio de localización a través del dispositivo GPS.	
Descripción: para tener una mejor precisión a la hora de ejecutar la localización se debe verificar que el mejor proveedor de localización sea el dispositivo GPS	
Responsable: Wilmer Barrera	**Fecha:** 27/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desarrollada la funcionalidad para verificar el mejor proveedor de localización - Tener activo el dispositivo GPS en el emulador	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar para verificar el mejor proveedor de localización - Verificar el resultado que se muestra en consola	
Resultado esperado: al buscar que el mejor proveedor de localización sea el dispositivo GPS, este proceso debe devolver una cedan de texto igual a "gps" que se imprimirá en la consola de la aplicación.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 02 de historia de usuario 12.

TAREA DE INGENIERÍA		
Id: TI_02	**Tipo de tarea:** desarrollo	**Sprint:** 7
Nombre de historia usuario: HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android		
Nombre tarea: Implementación de funcionalidad para usar el servicio de localización a través de la red celular a la que se encuentra conectado.		
Fecha inicio: 28/12/17		**Fecha fin:**28/12/17
Descripción: Como desarrollador necesito crear la funcionalidad en la cual me permita usar el servicio de localización a través de la red celular.		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_103	Verificar que el proveedor de localización de red esté activo
PA_104	Verificar que el proveedor de localización de red esté inactivo
PA_105	Verificar que el mejor proveedor de localización no sea GPS
PA_106	Verificar que el mejor proveedor de localización GPS esté inactivo

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 103.

PRUEBA DE ACEPTACIÓN	
Id: PA_103	**Nombre:** Verificar que el proveedor de localización de red esté activo
Tarea de ingeniería: TI_02 Implementación de funcionalidad para usar el servicio de localización a través de la red celular a la que se encuentra conectado.	
Descripción: cuando se necesita hacer una localización por el método de localización a través de la red, este se debe verificar que este proveedor se encuentre activo	
Responsable: Wilmer Barrera	**Fecha:** 28/12/17
Condición de ejecución: - Tener la última versión de la app instalada en el emulador - Tener desarrollada la funcionalidad para verificar el mejor proveedor de localización - Tener desarrollado la funcionalidad para activar el servicio de lo localización - Verificar que se encuentre activado los datos móviles - Verificar que se encuentre desactivado el dispositivo GPS	
Pasos de ejecución: - Ejecutar la aplicación desde el emulador - Presionar el botón para verificar el mejor proveedor de localización	

- Verificar la respuesta que se muestra en la consola.
Resultado esperado: cuando se requiere verificar que el mejor proveedor de localización es el de red, este proceso arroja un resultado en una cadena de texto igual a “network” que será presentado en pantalla en modo consola.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 104.

PRUEBA DE ACEPTACIÓN	
Id: PA_104	**Nombre:** Verificar que el proveedor de localización de red esté inactivo
Tarea de ingeniería: TI_02 Implementación de funcionalidad para usar el servicio de localización a través de la red celular a la que se encuentra conectado.	
Descripción: cuando se necesita hacer una localización por el método de localización a través de la red, este se debe verificar si este proveedor se encuentre inactivo, para poder advertir al usuario sobre este caso.	
Responsable: Wilmer Barrera	**Fecha:** 28/12/17
Condición de ejecución: - Tener la última versión de la app instalada en el emulador - Tener desarrollada la funcionalidad para verificar el mejor proveedor de localización - Tener desarrollado la funcionalidad para activar el servicio de lo localización - Verificar que se encuentre desactivado los datos móviles - Verificar que se encuentre desactivado el dispositivo GPS	
Pasos de ejecución: - Ejecutar la aplicación desde el emulador - Presionar el botón para verificar el mejor proveedor de localización - Verificar la respuesta que se muestra en la consola.	
Resultado esperado: cuando se requiere verificar que el proveedor de localización de red este inactivo este proceso arroja un resultado en una cadena de texto diferente a “network” que será presentado en pantalla en modo consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 105.

PRUEBA DE ACEPTACIÓN	
Id: PA_105	**Nombre:** Verificar que el mejor proveedor de localización no sea GPS
Tarea de ingeniería: TI_02 Implementación de funcionalidad para usar el servicio de localización a través de la red celular a la que se encuentra conectado.	
Descripción: cuando se necesita realizar una localización por el método de la red, se debe verificar que el mejor proveedor de localización no sea el dispositivo GPS	
Responsable: Wilmer Barrera	**Fecha:** 28/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desarrollada la funcionalidad para activar el servicio de localización - Tener desarrollada la funcionalidad para verificar el mejor proveedor - Tener desactivado el dispositivo GPS - Tener activado los datos móviles.	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador - Presionar el botón para obtener el mejor proveedor de localización activo - Verificar el resultado arrojado en consola	
Resultado esperado: el resultado en consola que se muestra después de ejecutar el proceso de verificación de mejor proveedor debe ser diferente a la cadena “gps” pudiendo ser la cadena “network” o “passive”.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 106.

PRUEBA DE ACEPTACIÓN	
Id: PA_106	**Nombre:** Verificar que el mejor proveedor de localización GPS esté inactivo
Tarea de ingeniería: TI_02 Implementación de funcionalidad para usar el servicio de localización a través de la red celular a la que se encuentra conectado.	
Descripción: cuando se necesita realizar una localización por el método de la red, se debe verificar que el proveedor de localización GPS esté inactivo.	
Responsable: Wilmer Barrera	**Fecha:** 28/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desarrollada la funcionalidad para activar el servicio de localización - Tener desarrollada la funcionalidad para verificar que el proveedor GPS está inactivo	

- Tener desactivado el dispositivo GPS
Pasos de ejecución: - Ejecutar la app móvil desde el emulador - Presionar el botón para obtener el mejor proveedor de localización activo - Verificar el resultado arrojado en consola
Resultado esperado: el resultado en consola que se muestra después de ejecutar el proceso de verificación si el dispositivo GPS está inactivo, este proceso devolverá un resultado booleano igual a false, lo que significa que el dispositivo GPS se encuentra desactivado.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla historia de usuario 13.

HISTORIA DE USUARIO		
Id: HU_13	**Nombre:** Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte	
Descripción: Como desarrollador necesito implementar una funcionalidad que me permita consultar los boletos que se encuentran disponibles en un horario de una ruta específica, la misma que funcionará con una conexión a internet de manera activa.		
Usuario: Wilmer Barrera	**Sprint:** 7	
Fecha inicio: 29/12/17	**Fecha fin:** 05/01/18	**Esfuerzo:** 25

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Implementar funcionalidad para verificar conexión a una red wifi, o por datos móviles
TI_02	Crear funcionalidad para consumo de servicios web mediante método post
TI_03	Crear funcionalidad para extracción y procesamiento de resultado devuelto por servicio web en hosting
TI_04	Creación de funcionalidad para ejecutar petición de boletos disponibles en segundo plano

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 13.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 7
Nombre de historia usuario: HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte		

Nombre tarea: Implementar funcionalidad para verificar conexión a una red wifi, o por datos móviles.	
Fecha inicio: 29/12/17	**Fecha fin:**29/12/17
Descripción: Con el fin de realizar una consulta de boletos disponibles a través del servicio web en el hosting, es necesario implementar una funcionalidad en la cual se verifique que existen una conexión ya sea por red wifi o datos móviles.	

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_107	Verificar si existe una conexión a internet activa
PA_108	Verificar que haya conexión a red activa wifi
PA_109	Verificar cuando no hay conexión a red wifi activa
PA_110	Verificar cuando hay conexión a red activa con datos móviles
PA_111	Verificar cuando no hay conexión a red activa con datos móviles
PA_112	Verificar activación de servicio para acceder a internet
PA_113	Verificar cuando no existe servicio activo para acceso a internet
PA_114	Verificar cuando existe conectividad y disponibilidad de red con datos móviles

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 107.

PRUEBA DE ACEPTACIÓN	
Id: PA_107	**Nombre:** Verificar si existe una conexión a internet activa
Tarea de ingeniería: TI_01 Implementar funcionalidad para verificar conexión a una red wifi, o por datos móviles.	
Descripción: para poder hacer una consulta de boletos se necesita verificar si se tiene una conexión a internet activa para poder llevar a cabo este proceso.	
Responsable: Wilmer Barrera	**Fecha:** 29/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener declarado el permiso de acceso a internet en la aplicación - Tener desarrollada la funcionalidad para controlar si existe una conexión activa a internet. - Tener activado una conexión a internet ya sea una conexión a través de la red wifi o datos móviles	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador	

- Presionar el botón donde verifica si existe una conexión a internet activa - Verificar el resultado arrojado en la consola
Resultado esperado: cuando se verifica si existe una conexión a internet activa ya sea a través de datos móviles o una conexión a una red Wi Fi, este proceso devolverá un valor booleano true lo que se mostrará como un mensaje de confirmación dando a entender que si existe conexión activa a internet.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 108.

PRUEBA DE ACEPTACIÓN	
Id: PA_108	**Nombre:** Verificar que haya conexión a red activa wifi
Tarea de ingeniería: TI_01 Implementar funcionalidad para verificar conexión a una red wifi, o por datos móviles.	
Descripción: para poder verificar si se está conectado a internet es necesario verificar si la conexión a través de wifi esté activa.	
Responsable: Wilmer Barrera	**Fecha:** 29/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener activado la conexión a una red wifi - Tener desarrollado la funcionalidad para verificar una conexión a internet.	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador - Verificar que el cuadro de dialogo con la respuesta de conexión a red wifi	
Resultado esperado: cuando se verifica que exista una conexión a una red wifi este proceso devolverá como resultado una cadena de texto igual a “wifi”, lo que significa que si hay una conexión activa a través de wifi	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 5s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 109.

PRUEBA DE ACEPTACIÓN	
Id: PA_109	**Nombre:** Verificar cuando no hay conexión a red wifi activa
Tarea de ingeniería: TI_01 Implementar funcionalidad para verificar conexión a una red wifi, o por datos móviles.	

Descripción: para poder verificar si se está conectado a internet y no existe una conexión a través de wifi esté activa. Esta debe ser controlada en la app	
Responsable: Wilmer Barrera	**Fecha:** 29/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desactivado la conexión a una red wifi - Tener desarrollado la funcionalidad para verificar una conexión a internet.	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador - Verificar que el cuadro de dialogo con la respuesta de conexión a red wifi	
Resultado esperado: cuando se verifica si no existe una conexión a una red wifi este proceso devolverá como resultado una cadena de texto diferente a "wifi", lo que significa que no se tiene una conexión activa a través de wifi	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 110.

PRUEBA DE ACEPTACIÓN	
Id: PA_110	**Nombre:** Verificar cuando hay conexión a red activa con datos móviles
Tarea de ingeniería: TI_01 Implementar funcionalidad para verificar conexión a una red wifi, o por datos móviles.	
Descripción: para poder verificar si se está conectado a internet es necesario verificar si hay una conexión a internet a través de red de datos móviles esté activa.	
Responsable: Wilmer Barrera	**Fecha:** 29/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desactivado la conexión a una red wifi - Tener activado la conexión a internet por datos móviles - Tener desarrollado la funcionalidad para verificar una conexión a internet.	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador - Verificar que el cuadro de dialogo con la respuesta de conexión a red de datos móviles	
Resultado esperado: cuando se verifica que exista una conexión a una red de datos móviles este proceso devolverá como resultado una cadena de texto igual a "mobile", lo que significa que si hay una conexión activa a través de datos móviles	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 5s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 111.

PRUEBA DE ACEPTACIÓN	
Id: PA_111	**Nombre:** Verificar cuando no hay conexión a red activa con datos móviles
Tarea de ingeniería: TI_01 Implementar funcionalidad para verificar conexión a una red wifi, o por datos móviles.	
Descripción: para poder verificar si se está conectado a internet es necesario verificar cuando no hay una conexión a internet a través de red de datos móviles que esté activa.	
Responsable: Wilmer Barrera	**Fecha:** 29/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desactivado la conexión a una red wifi - Tener desactivado la conexión a internet por datos móviles - Tener desarrollado la funcionalidad para verificar una conexión a internet.	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador - Verificar que el cuadro de dialogo con la respuesta de conexión a red de datos móviles	
Resultado esperado: cuando se verifica que no exista una conexión a una red de datos móviles este proceso devolverá como resultado una cadena de texto diferente a "mobile", lo que significa que no hay una conexión activa a través de datos móviles	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 112.

PRUEBA DE ACEPTACIÓN	
Id: PA_112	**Nombre:** Verificar activación de servicio para acceder a internet
Tarea de ingeniería: TI_01 Implementar funcionalidad para verificar conexión a una red wifi, o por datos móviles.	
Descripción: para poder usar el servicio de internet es necesario verificar que se active este servicio antes de usarlo.	
Responsable: Wilmer Barrera	**Fecha:** 29/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener una conexión a internet a través de datos móviles o red wifi - Tener desarrollada la funcionalidad para activar el acceso a internet	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador	

- Presionar botón para activar el acceso a internet - Verificar la respuesta que muestra la consola
Resultado esperado: cuando se requiera activar el servicio de internet para tener acceso al mismo, este debe devolver un resultado con un valor booleano igual a true, lo que significa que se activó el servicio de internet.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 113.

PRUEBA DE ACEPTACIÓN	
Id: PA_113	**Nombre:** Verificar cuando no existe servicio activo para acceso a internet
Tarea de ingeniería: TI_01 Implementar funcionalidad para verificar conexión a una red wifi, o por datos móviles.	
Descripción: para poder usar el servicio de internet es necesario verificar incluso cuando no se encuentra activo este servicio. Para de eta manera poder controlar este comportamiento en la app móvil.	
Responsable: Wilmer Barrera	**Fecha:** 29/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener una desactivada la conexión a internet a través de datos móviles y red wifi - Tener desarrollada la funcionalidad para verificar la activación del acceso a internet	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador - Presionar botón para activar el acceso a internet - Verificar la respuesta que muestra la consola	
Resultado esperado: cuando se requiera verificar que no exista un servicio activo para internet, este proceso debe devolver un resultado igual a null, lo que significa que el servicio se encuentra desactivado.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 114.

PRUEBA DE ACEPTACIÓN	
Id: PA_114	**Nombre:** Verificar cuando existe conectividad y disponibilidad de red con datos móviles
Tarea de ingeniería: TI_01 Implementar funcionalidad para verificar conexión a una red wifi, o por datos móviles.	
Descripción: para poder usar el servicio de internet se debe verificar que exista la disponibilidad de red y la conectividad de la misma.	
Responsable: Wilmer Barrera	**Fecha:** 29/12/17
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener implementada la funcionalidad para verificar la conectividad y disponibilidad de la red para acceso a internet. - Tener activa la conexión a la red a través de wifi o datos móviles	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador - Presionar el botón para verificar la disponibilidad y conectividad del acceso a la red - Verificar la respuesta que se muestra en la consola.	
Resultado esperado: para verificar la conectividad y disponibilidad a la red este proceso devolverá un valor entero igual a 2, que se imprimirá en la consola dando a entender que si existe conectividad y disponibilidad.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 02 de historia de usuario 13.

TAREA DE INGENIERÍA		
Id: TI_02	**Tipo de tarea:** desarrollo	**Sprint:** 7
Nombre de historia usuario: HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte		
Nombre tarea: Crear funcionalidad para consumo de servicios web mediante método post.		
Fecha inicio: 02/01/18		**Fecha fin:**03/01/18
Descripción: Para realizar peticiones al servicio web del servicio de hosting se realizará mediante una petición http con el método post, por lo que se implementará dicha funcionalidad.		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	

PA_115	Verificar si se abrió una petición http de consumo de servicio web exitoso
PA_116	Verificar si el estado de petición fue aceptado
PA_117	Verificar si la petición fue rechazada
PA_118	Verificar que el resultado devuelto a petición no sea vacío
PA_119	Verificar cuando no hay resultado de servicio web

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 115.

PRUEBA DE ACEPTACIÓN	
Id: PA_115	**Nombre:** Verificar si se abrió una petición http de consumo de servicio web exitoso
Tarea de ingeniería: TI_02 Crear funcionalidad para consumo de servicios web mediante método post	
Descripción: cuando se abre una petición para consultar la lista de boletos disponibles esta petición se debe abrir correctamente con la url correcta, por lo que se necesita verificar esta situación.	
Responsable: Wilmer Barrera	**Fecha:** 03/01/18
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener una conexión a internet activa - Tener desarrollada la funcionalidad para abrir una petición http	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador - Presionar el botón para abrir la petición de consumo de servicio web de consulta de boletos disponibles. - Verificar la respuesta que se muestra en la consola	
Resultado esperado: cuando se abre una petición http para consultar la lista de boletos disponibles, este proceso devolverá un valor booleano igual a true lo que significa que la petición se abrió correctamente.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 116.

PRUEBA DE ACEPTACIÓN	
Id: PA_116	**Nombre:** Verificar si el estado de petición fue aceptado
Tarea de ingeniería: TI_02 Crear funcionalidad para consumo de servicios web mediante método post	
Descripción: cuando se realiza una petición para consultar la lista de boletos disponibles esta petición primeramente debe ser aceptada, por lo que se necesita verificar esta situación.	
Responsable: Wilmer Barrera	**Fecha:** 03/01/18
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener una conexión a internet activa - Tener desarrollada la funcionalidad para verificar el estado de la petición http	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador - Presionar el botón para realizar petición de consumo de servicio web de consulta de boletos disponibles. - Verificar la respuesta que se muestra en la consola	
Resultado esperado: cuando se realiza la petición http para consultar la lista de boletos disponibles esta debe ser aceptada, este proceso devolverá un valor booleano igual a true lo que significa que la petición fue aceptada.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 117.

PRUEBA DE ACEPTACIÓN	
Id: PA_117	**Nombre:** Verificar si la petición fue rechazada
Tarea de ingeniería: TI_02 Crear funcionalidad para consumo de servicios web mediante método post	
Descripción: cuando se realiza una petición para consultar la lista de boletos disponibles si esta petición es rechazada se debe controlar esta situación.	
Responsable: Wilmer Barrera	**Fecha:** 03/01/18
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desactivada la conexión a internet - Tener desarrollada la funcionalidad para verificar el estado de la petición http	
Pasos de ejecución:	

- Ejecutar la app móvil desde el emulador - Presionar el botón para realizar petición de consumo de servicio web de consulta de boletos disponibles. - Verificar la respuesta que se muestra en la consola
Resultado esperado: cuando se realiza la petición http para consultar la lista de boletos disponibles y esta es rechazada, este proceso devolverá un valor booleano igual a false lo que significa que la petición fue rechazada
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 118.

PRUEBA DE ACEPTACIÓN	
Id: PA_118	**Nombre:** Verificar que el resultado devuelto a petición no sea vacío
Tarea de ingeniería: TI_02 Crear funcionalidad para consumo de servicios web mediante método post	
Descripción: cuando se realiza la petición para consulta de consulta de boletos disponibles se debe verificar que esta petición no dé una respuesta vacía.	
Responsable: Wilmer Barrera	**Fecha:** 03/01/18
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desarrollada la funcionalidad para realizar petición de boletos disponibles - Tener activa una conexión a internet ya sea por datos móviles o red wifi	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador - Presionar el botón para consultar boletos disponibles de un transporte - Verificar el tipo de respuesta que muestra la consola de la app	
Resultado esperado: cuando se realiza la petición para consultar los boletos disponibles de un transporte y existe una respuesta desde este servicio entonces este proceso devolverá una respuesta diferente a nulo. Imprimiendo así el texto plano de respuesta.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 8s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 119.

PRUEBA DE ACEPTACIÓN	
Id: PA_119	**Nombre:** Verificar cuando no hay resultado de servicio web
Tarea de ingeniería: TI_02 Crear funcionalidad para consumo de servicios web mediante método post	
Descripción: cuando se realiza la petición para consulta de consulta de boletos disponibles y no existe resultado de este, antes este caso el sistema debe reaccionar para advertir al usuario.	
Responsable: Wilmer Barrera	**Fecha:** 03/01/18
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desarrollada la funcionalidad para realizar petición de boletos disponibles - Tener activa una conexión a internet ya sea por datos móviles o red wifi	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador - Presionar el botón para consultar boletos disponibles de un transporte - Verificar el tipo de respuesta que muestra la consola de la app	
Resultado esperado: cuando se realiza la petición para consultar los boletos disponibles de un transporte y no existe una respuesta desde este servicio entonces este proceso devolverá una respuesta vacía (“”). Este resultado será advertido a través de un mensaje en pantalla.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 7s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 03 de historia de usuario 13.

TAREA DE INGENIERÍA		
Id: TI_03	**Tipo de tarea:** desarrollo	**Sprint:** 7
Nombre de historia usuario: HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte		
Nombre tarea: Crear funcionalidad para extracción y procesamiento de resultado devuelto por servicio web en hosting.		
Fecha inicio: 04/01/18		**Fecha fin:**04/01/18
Descripción: Para poder presentar información entendible al usuario se procederá a crear la funcionalidad que permita extraer los resultados del consumo del servicio web y así como también procesar esta información para luego de esto mostrarla al usuario		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	

PA_120	Verificar la disponibilidad de asientos desde respuesta de servicio web
PA_121	Verificar que el resultado del servicio web sea una consulta correcta
PA_122	Verificar que el resultado del servicio web sea una consulta incorrecta
PA_123	Verificar extracción de número de asientos disponibles
PA_124	Verificar extracción de asientos ocupados sea correcto

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 120.

PRUEBA DE ACEPTACIÓN	
Id: PA_120	**Nombre:** Verificar la disponibilidad de asientos desde respuesta de servicio web
Tarea de ingeniería: TI_03 Crear funcionalidad para extracción y procesamiento de resultado devuelto por servicio web en hosting.	
Descripción: cuando se realice una petición de consulta de boletos disponibles, si existe una respuesta desde el servicio web esta deberá ser verificada.	
Responsable: Wilmer Barrera	**Fecha:** 04/01/18
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desarrollada la conexión con los servicios web - Tener desarrollada datos en el hosting sobre los boletos disponibles - Tener acceso a internet desde datos móviles o red wifi	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador - Presionar el botón para realizar petición de consulta de boletos disponibles. - Verificar la respuesta que se trae desde el servicio web	
Resultado esperado: cuando se realiza una consulta de boletos disponibles de una ruta de transporte, este debe verificarse que traiga respuesta, en caso de tener respuesta esta respuesta será diferente de nula, la cual se imprimirá en la consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 27s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 121.

PRUEBA DE ACEPTACIÓN	
Id: PA_121	**Nombre:** Verificar que el resultado del servicio web sea una consulta correcta
Tarea de ingeniería: TI_03 Crear funcionalidad para extracción y procesamiento de resultado devuelto por servicio web en hosting.	

Descripción: cuando se realiza una petición de consulta de boletos disponibles este devolverá un resultado en un array de objetos JSon que se deberá verificar si trae resultados de una consulta correcta.	
Responsable: Wilmer Barrera	**Fecha:** 04/01/18
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desarrollado la conexión con el servicio web - Tener acceso a internet a través de datos móviles o red wifi - Tener desarrollada la funcionalidad para realizar petición de consulta de boletos disponibles	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador - Presionar el botón para consultar boletos disponibles - Verificar es estado de la respuesta extraída desde el servicio web	
Resultado esperado: cuando se extraiga el estado de la respuesta Json, si el estado viene con una respuesta de consulta correcta mostrará una cadena de texto igual a "CC", lo que significa que trae una respuesta de una consulta correcta.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 122.

PRUEBA DE ACEPTACIÓN	
Id: PA_122	**Nombre:** Verificar que el resultado del servicio web sea una consulta incorrecta
Tarea de ingeniería: TI_03 Crear funcionalidad para extracción y procesamiento de resultado devuelto por servicio web en hosting.	
Descripción: cuando se realiza una petición de consulta de boletos disponibles y no se ha obtenido resultado de una consulta este devolverá un resultado en un objeto JSon y se deberá verificar si el estado de esta respuesta.	
Responsable: Wilmer Barrera	**Fecha:** 04/01/18
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desarrollado la conexión con el servicio web - Tener acceso a internet a través de datos móviles o red wifi - Tener desarrollada la funcionalidad para realizar petición de consulta de boletos disponibles	
Pasos de ejecución:	

- Ejecutar la app móvil desde el emulador - Presionar el botón para consultar boletos disponibles - Verificar es estado de la respuesta extraída desde el servicio web
Resultado esperado: cuando se extraiga el estado de la respuesta Json, si el estado viene con una respuesta de consulta sin éxito o resultado mostrará una cadena de texto igual a "CI", lo que significa que trae una respuesta de una consulta insatisfactoria.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 123.

PRUEBA DE ACEPTACIÓN	
Id: PA_123	**Nombre:** Verificar extracción de número de asientos disponibles
Tarea de ingeniería: TI_03 Crear funcionalidad para extracción y procesamiento de resultado devuelto por servicio web en hosting.	
Descripción: para verificar la lista de asientos disponibles de un transporte en una ruta y horario específico este proceso se debe extraer solo los asientos disponibles.	
Responsable: Wilmer Barrera	**Fecha:** 04/01/18
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener acceso a internet - Tener datos en la lista de boletos disponibles en el hosting - Tener una conexión con el servicio web alojado en el hosting	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador - Presionar el botón que consume la consulta de boletos disponibles de una unidad de transporte, esto basado en el horario y la ruta. - Verificar la lista de asientos disponibles imprimidos en consola	
Resultado esperado: en la respuesta que se extrae de servicio web cuando se realiza la consulta de boletos disponibles, este proceso trae la lista de asientos ocupados y genera la lista de asientos disponibles de acuerdo a la cantidad máxima de asientos que existe en ese transporte, este resultado se presenta en pantalla en consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 63s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 124.

PRUEBA DE ACEPTACIÓN	
Id: PA_124	**Nombre:** Verificar extracción de asientos ocupados sea correcto
Tarea de ingeniería: TI_03 Crear funcionalidad para extracción y procesamiento de resultado devuelto por servicio web en hosting.	
Descripción: cuando se realiza la consulta de boletos disponibles, se debe verificar la cantidad de boletos que se encuentran disponibles para de esta manera informar al usuario.	
Responsable: Wilmer Barrera	**Fecha:** 04/01/18
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener una conexión activa a internet - Tener desarrollada la funcionalidad de consulta de boletos disponibles en un servicio web	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador - Presionar el botón para consulta de servicios web disponibles - Verificar el tipo de respuesta que se imprime en la consola	
Resultado esperado: este proceso deberá contar cada uno de los asientos ocupados y mostrar este resultado como un número entero que se mostrará en consola.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 04 de historia de usuario 13.

TAREA DE INGENIERÍA		
Id: TI_04	**Tipo de tarea:** desarrollo	**Sprint:** 7
Nombre de historia usuario: HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte		
Nombre tarea: Creación de funcionalidad para ejecutar petición de boletos disponibles en segundo plano.		
Fecha inicio: 05/01/18		**Fecha fin:**05/01/18
Descripción: Con el fin de que la aplicación no entre a un estado de no responder, las peticiones de consulta de boletos disponibles se realizarán en segundo plano.		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_125	Verificar que ejecución de consumo de servicio web en segundo plano.	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 125.

PRUEBA DE ACEPTACIÓN	
Id: PA_125	**Nombre:** Verificar que ejecución de consumo de servicio web en segundo plano.
Tarea de ingeniería: TI_04 Creación de funcionalidad para ejecutar petición de boletos disponibles en segundo plano.	
Descripción: Para el consumo del servicio web es necesario que esta petición se ejecute en segundo plano, esto con el fin de que el sistema operativo no asuma que la aplicación se detuvo esperando una respuesta ejecutada en el hilo principal de la app	
Responsable: Wilmer Barrera	**Fecha:** 05/01/18
Condición de ejecución: - Tener creada la conexión al servicio web - Tener instalada la última versión de la app en el emulador	
Pasos de ejecución: - Ejecutar el emulador con la aplicación - Esperar 6 segundos para verificar que la aplicación no se detenga - Verificar que se pueda seguir usando la aplicación normalmente.	
Resultado esperado: al ejecutar la petición de consumo del servicio web en segundo plano este no detendrá la aplicación en caso de tardarse en traer una respuesta, así pase más de 5 segundos.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 7s

Realizado por: Wilmer B. 2018.

Sprint 8

Tabla historia de usuario 14.

HISTORIA DE USUARIO		
Id: HU_14	**Nombre:** Implementación de funcionalidad favoritos	
Descripción: Como desarrollador necesito implementar la funcionalidad que me permita agregar, visualizar aquellas rutas favoritas.		
Usuario: Wilmer Barrera	**Sprint:** 8	
Fecha inicio: 10/01/18	**Fecha fin:** 16/01/18	**Esfuerzo:** 25

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Implementar funcionalidad para extraer lista de favoritos

TI_02	Implementar funcionalidad parar extraer lista de objetos favoritos a lista de cadena de string
TI_03	Implementar funcionalidad para agregar una ruta favorita a la lista
TI_04	Implementar funcionalidad para extraer detalles de una ruta favorita en base a su id

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 14.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 8
Nombre de historia usuario: HU_14 Implementación de funcionalidad favoritos		
Nombre tarea: Implementar funcionalidad para extraer lista de favoritos.		
Fecha inicio: 10/01/18		**Fecha fin:**10/01/18
Descripción: Como desarrollador necesito crear la funcionalidad en la que se extraerán la lista de rutas favoritas desde la base de datos local		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_126	Verificar que se cargue lista de rutas favoritas desde la base de datos local	
PA_127	Verificar cuando la lista de rutas favoritas esté vacía	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 126.

PRUEBA DE ACEPTACIÓN	
Id: PA_126	**Nombre:** Verificar que se cargue lista de rutas favoritas desde la base de datos local
Tarea de ingeniería: TI_01 Implementar funcionalidad para extraer lista de favoritos	
Descripción: para poder mostrar la lista de favoritos que se tienen registrados, la extracción debe ser controlada si fue extraída con éxito	
Responsable: Wilmer Barrera	**Fecha:** 10/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla FAVORITO con datos - Tener creada la conexión a la base de datos - Tener creada la funcionalidad para extracción de datos de la tabla favoritos.	
Pasos de ejecución:	

- Ejecutar la aplicación desde el emulador - Presionar botón para extraer lista de favoritos desde la base de datos. - Verificar la respuesta extraída que se muestra en el emulador
Resultado esperado: cuando se extrae la lista de datos de la tabla favoritos, este proceso devolverá una lista de objetos favoritos con un número entero de elementos, cuando trae respuesta de la consulta este número de elementos es imprimido en la consola, siendo este mayor a 0
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 127.

PRUEBA DE ACEPTACIÓN	
Id: PA_127	**Nombre:** Verificar cuando la lista de rutas favoritas esté vacía
Tarea de ingeniería: TI_01 Implementar funcionalidad para extraer lista de favoritos	
Descripción: para poder mostrar la lista de favoritos que se tienen registrados, es necesario verificar cuando la extracción no trae resultados.	
Responsable: Wilmer Barrera	**Fecha:** 10/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla FAVORITO sin datos - Tener creada la conexión a la base de datos - Tener creada la funcionalidad para extracción de datos de la tabla favoritos.	
Pasos de ejecución: - Ejecutar la aplicación desde el emulador - Presionar botón para extraer lista de favoritos desde la base de datos. - Verificar la respuesta extraída que se muestra en el emulador	
Resultado esperado: cuando se extrae la lista de datos de la tabla favoritos, este proceso devolverá una lista de objetos favoritos con un número entero de cero elementos, este valor se imprimirá en la consola de la app.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 02 de historia de usuario 14.

TAREA DE INGENIERÍA		
Id: TI_02	**Tipo de tarea:** desarrollo	**Sprint:** 8
Nombre de historia usuario: HU_14 Implementación de funcionalidad favoritos		
Nombre tarea: Implementar funcionalidad parar extraer lista de objetos favoritos a lista de cadena de string.		
Fecha inicio: 11/01/18		**Fecha fin:**11/01/18
Descripción: Como desarrollador necesito crear la funcionalidad con la cual me permita pasar una lista de objetos de las rutas favoritas a una lista de strings, esto para posteriormente ser presentado al usuario.		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_128	Verificar formateo de lista de objetos con rutas favoritas a lista de strings para presentación de estos.	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 128.

PRUEBA DE ACEPTACIÓN	
Id: PA_128	**Nombre:** Verificar formateo de lista de objetos con rutas favoritas a lista de strings para presentación de estos
Tarea de ingeniería: TI_02 Implementar funcionalidad parar extraer lista de objetos favoritos a lista de cadena de string.	
Descripción: antes de mostrar la lista de favoritos al usuario se debe verificar la adaptación de la información de la lista de objetos a la lista de strings.	
Responsable: Wilmer Barrera	**Fecha:** 11/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla FAVORITO con datos - Tener creada la conexión a la base de datos - Tener creada la funcionalidad para extracción de datos de la tabla favoritos. - Tener creada a funcionalidad para convertir lista de objetos en lista de strings	
Pasos de ejecución: - Ejecutar la aplicación desde el emulador - Presionar botón para extraer y convertir lista de favoritos a lista de strings - Verificar la respuesta extraída que se muestra en el emulador	

Resultado esperado: al convertir la lista de objetos favoritos en lista de strings cada objeto debe tener el siguiente formato: La ruta la cual es la ciudad origen y la ciudad destino separados por un símbolo slash "/" seguido por la hora de salida ejemplo: "Quito/Guayaquil 22:55"	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 16s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 03 de historia de usuario 14.

TAREA DE INGENIERÍA		
Id: TI_03	**Tipo de tarea:** desarrollo	**Sprint:** 8
Nombre de historia usuario: HU_14 Implementación de funcionalidad favoritos		
Nombre tarea: Implementar funcionalidad para agregar una ruta favorita a la lista.		
Fecha inicio: 12/01/18		**Fecha fin:**12/01/18
Descripción: Como desarrollador necesito crear la funcionalidad en la que permita al usuario agregar una ruta favorita al usuario		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_129	Verificar si dicha ruta favorita ya existe
PA_130	Verificar si una ruta fue agregada correctamente
PA_131	Verificar si una ruta específica que se encuentra dentro de favoritos

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 129.

PRUEBA DE ACEPTACIÓN	
Id: PA_129	**Nombre:** Verificar si dicha ruta favorita ya existe
Tarea de ingeniería: TI_03 Implementar funcionalidad para agregar una ruta favorita a la lista	
Descripción: antes de agregar una nueva ruta como favorita es necesario verificar que no exista agregado ese mismo favorito.	
Responsable: Wilmer Barrera	**Fecha:** 12/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla FAVORITO con datos - Tener creada la conexión a la base de datos - Tener creada la funcionalidad para verificar que no exista una ruta igual registrada en favoritos	
Pasos de ejecución:	

- Ejecutar la aplicación desde el emulador - Presionar botón para verificar la existencia de un favorito - Verificar la respuesta extraída que se muestra en el emulador
Resultado esperado: cuando se verifique que una determinada ruta ya existe registrada en la tabla favoritos, este proceso devolverá un valor booleano igual a true, lo que significa que si existe dicha ruta ya agregada.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 130.

PRUEBA DE ACEPTACIÓN	
Id: PA_130	**Nombre:** Verificar si una ruta fue agregada correctamente
Tarea de ingeniería: TI_03 Implementar funcionalidad para agregar una ruta favorita a la lista	
Descripción: cuando se agrega una ruta como favorita, se debe verificar que esta ruta sea agregada correctamente.	
Responsable: Wilmer Barrera	**Fecha:** 12/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla FAVORITO - Tener instalada la última versión de la app en el emulador. - Tener creada funcionalidad para agregar ruta a favoritos.	
Pasos de ejecución: - Ejecutar la app desde el emulador - Ingresar al horario de una ruta específica - Ver detalle de dicha ruta - Presionar botón para agregar a favoritos - Verificar el resultado que sobre el estado de haber agregado una ruta a favorito	
Resultado esperado: cuando se agrega una ruta específica q la tabla favoritos este proceso devolverá una respuesta booleana igual al insertarse correctamente dicho registro.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 131.

PRUEBA DE ACEPTACIÓN	
Id: PA_131	**Nombre:** Verificar si una ruta específica que se encuentra dentro de favoritos
Tarea de ingeniería: TI_03 Implementar funcionalidad para agregar una ruta favorita a la lista	

Descripción: cuando se requiere recuperar una ruta específica de la lista de rutas favoritas se debe verificar que esta ruta se encuentre registrada.	
Responsable: Wilmer Barrera	**Fecha:** 12/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla FAVORITO con datos - Tener creada la conexión a la base de datos - Tener creada la funcionalidad para presentar la lista de rutas favoritas.	
Pasos de ejecución: - Ingresar parámetro de búsqueda, este parámetro es el id de la ruta y la hora de salida - Ejecutar la app desde el emulador - Presionar el botón para buscar recuperar una ruta con un mismo horario agregado. - Verificar el resultado presentado en consola.	
Resultado esperado: cuando se requiere verificar una determinada ruta y no se encuentra en registrada, este proceso devolverá una respuesta nula, lo que significa que no existe registrada dicha ruta.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 04 de historia de usuario 14.

TAREA DE INGENIERÍA		
Id: TI_04	**Tipo de tarea:** desarrollo	**Sprint:** 8
Nombre de historia usuario: HU_14 Implementación de funcionalidad favoritos		
Nombre tarea: Implementar funcionalidad para extraer detalles de una ruta favorita en base a su id.		
Fecha inicio: 15/01/18		**Fecha fin:**16/01/18
Descripción: Como desarrollador necesito crear la funcionalidad en la que se extraerán una ruta favorita y sus detalles desde la base de datos local		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_132	Verificar extracción de una ruta favorita en base a su id y hora

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 132.

PRUEBA DE ACEPTACIÓN	
Id: PA_132	**Nombre:** Verificar extracción de una ruta favorita en base a su id y hora
Tarea de ingeniería: TI_04 Implementar funcionalidad para extraer detalles de una ruta favorita en base a su id.	
Descripción: cuando se requiere recuperar una ruta específica de la lista de rutas favoritas se debe verificar que esta consulta traiga resultados.	
Responsable: Wilmer Barrera	**Fecha:** 16/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla FAVORITO con datos - Tener creada la conexión a la base de datos - Tener creada la funcionalidad para presentar la lista de rutas favoritas.	
Pasos de ejecución: - Ingresar los parámetros de búsqueda, estos parámetros son el id de la ruta y la hora de salida. - Ejecutar la app desde el emulador - Presionar el botón para buscar recuperar una ruta con un mismo horario agregado. - Verificar el resultado presentado en consola.	
Resultado esperado: cuando se requiere recuperar una determinada ruta y se encuentra registrada, este proceso devolverá una respuesta diferente de nulo.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla historia de usuario 15.

HISTORIA DE USUARIO		
Id: HU_15	**Nombre:** Implementación de funcionalidad llamada telefónica y cargar datos de agencia de cooperativa	
Descripción: Como desarrollador necesito implementar una funcionalidad que me permita mostrar los datos de una agencia de una cooperativa de transporte específica, así como poder contactarme con esta a través de una llamada telefónica.		
Usuario: Wilmer Barrera	**Sprint:** 8	
Fecha inicio: 17/01/18	**Fecha fin:** 19/01/18	**Esfuerzo:** 15

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Implementar funcionalidad para extraer datos de una agencia de transporte

TI_02	Implementar funcionalidad para realizar una llamada telefónica desde la aplicación

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 15.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 8
Nombre de historia usuario: HU_15 Implementación de funcionalidad llamada telefónica y cargar datos de agencia de cooperativa		
Nombre tarea: Implementar funcionalidad para extraer datos de una agencia de transporte.		
Fecha inicio: 17/01/18		**Fecha fin:**18/01/18
Descripción: Como desarrollador necesito crear la funcionalidad en la que se extraerán los datos de una agencia de transporte desde la base de datos local		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_133	Verificar extracción de datos de una agencia de cooperativa	
PA_134	Verificar cuando no existe agencia o datos de agencia de una cooperativa de transporte	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 133.

PRUEBA DE ACEPTACIÓN	
Id: PA_133	**Nombre:** Verificar extracción de datos de una agencia de cooperativa
Tarea de ingeniería: TI_01 Implementar funcionalidad para extraer datos de una agencia de transporte	
Descripción: cuando se extraiga la información de una agencia de cooperativa se debe verificar si esta consulta devuelve un resultado.	
Responsable: Wilmer Barrera	**Fecha:** 18/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla COOPERATIVA con datos - Tener creada la conexión con la base de datos - Tener creada la funcionalidad para extraer la información de una agencia	
Pasos de ejecución: - Ingresar el parámetro de búsqueda de agencia, este es el id de una agencia determinada - Ejecutar la app desde el emulador	

- Presionar botón para cargar la información de agencia de cooperativa
Resultado esperado: cuando se mande a cargar los datos de una cooperativa determinada este proceso debe devolverá la información en un objeto Dependencia, este objeto será diferente de null, y la información del objeto será imprimida en modo consola.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** 14s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 134.

PRUEBA DE ACEPTACIÓN	
Id: PA_134	**Nombre:** Verificar cuando no existe agencia o datos de agencia de una cooperativa de transporte
Tarea de ingeniería: TI_01 Implementar funcionalidad para extraer datos de una agencia de transporte	
Descripción: cuando se extraiga la información de una agencia de cooperativa se debe verificar en el caso de que no exista resultado devuelto por esta consulta.	
Responsable: Wilmer Barrera	**Fecha:** 18/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla COOPERATIVA sin datos - Tener creada la conexión con la base de datos - Tener creada la funcionalidad para extraer la información de una agencia	
Pasos de ejecución: - Ingresar el parámetro de búsqueda de agencia, este es el id de una agencia determinada - Ejecutar la app desde el emulador - Presionar botón para cargar la información de agencia de cooperativa	
Resultado esperado: cuando se mande a cargar los datos de una cooperativa determinada este proceso debe devolverá la información en un objeto Dependencia nulo cuando no se ha encontrado resultado.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 02 de historia de usuario 15.

TAREA DE INGENIERÍA		
Id: TI_02	**Tipo de tarea:** desarrollo	**Sprint:** 8
Nombre de historia usuario: HU_15 Implementación de funcionalidad llamada telefónica y cargar datos de agencia de cooperativa		
Nombre tarea: Implementar funcionalidad para realizar una llamada telefónica desde la aplicación.		
Fecha inicio: 18/01/18		**Fecha fin:**19/01/18
Descripción: Con el fin de contactarse con una agencia específica de cooperativa de transporte, se implementará la funcionalidad que permita realizar una llamada telefónica directamente desde la aplicación móvil		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_135	Verificar que permiso de llamada se encuentre activo	
PA_136	Verificar cuando no existe permiso de llamada activo	
PA_137	Verificar conversión de número en formato texto a formato numérico para llamar	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 135.

PRUEBA DE ACEPTACIÓN	
Id: PA_135	**Nombre:** Verificar que permiso de llamada se encuentre activo
Tarea de ingeniería: TI_02 Implementar funcionalidad para realizar una llamada telefónica desde la aplicación.	
Descripción: cuando se requiere hacer una llamada es necesario verificar si se tiene el permiso concedido por el usuario.	
Responsable: Wilmer Barrera	**Fecha:** 19/01/18
Condición de ejecución: - Tener declarado el permiso de llamada en la aplicación - Tener instalada la última versión de la app en el emulador - Tener desarrollada la funcionalidad para llamar	
Pasos de ejecución: - Ejecutar app desde el emulador - Presionar botón para verificar si existe permiso de llamada concedido. - Verificar tipo de respuesta imprimida en la consola.	

Resultado esperado: cuando se manda a verificar si existe el permiso activo de llamada, este proceso seguirá ejecutándose normalmente e imprimirá un mensaje indicando que es posible hacer la llamada.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 136.

PRUEBA DE ACEPTACIÓN	
Id: PA_136	**Nombre:** Verificar cuando no existe permiso de llamada activo
Tarea de ingeniería: TI_02 Implementar funcionalidad para realizar una llamada telefónica desde la aplicación.	
Descripción: cuando se requiere hacer una llamada es necesario verificar cuando no se tiene el permiso concedido por el usuario.	
Responsable: Wilmer Barrera	**Fecha:** 19/01/18
Condición de ejecución: - Tener declarado el permiso de llamada en la aplicación - Tener instalada la última versión de la app en el emulador - Tener desarrollada la funcionalidad para llamar	
Pasos de ejecución: - Ejecutar app desde el emulador - Presionar botón para verificar si existe permiso de llamada concedido. - Verificar tipo de respuesta imprimida en la consola.	
Resultado esperado: cuando se manda a verificar si no existe el permiso activo de llamada, este proceso mostrará un cuadro de dialogo en el que advierte que se necesita conceder permiso para realizar una llamada, dando la opción de conceder permiso o rechazar el mismo.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 137.

PRUEBA DE ACEPTACIÓN	
Id: PA_137	**Nombre:** Verificar conversión de número en formato texto a formato numérico para llamar
Tarea de ingeniería: TI_02 Implementar funcionalidad para realizar una llamada telefónica desde la aplicación.	
Descripción: para poder realizar llamada telefónica es necesario convertir un número en formato cadena a formato Uri, caso contrario no se podrá realizar dicha llamada y ocurrirá una excepción.	
Responsable: Wilmer Barrera	**Fecha:** 19/01/18
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desarrollada la funcionalidad para realizar llamada telefónica - Tener declarada el permiso para llamada en la app	
Pasos de ejecución: - Ingresar el parámetro para realizar llamada telefónica, este parámetro es el número telefónico - Ejecutar la app desde el emulador - Presionar el botón de llamada telefónica - Verificar la pantalla si presenta una acción de llamada saliente	
Resultado esperado: cuando se presiona el botón para realizar una llamada telefónica esta se llevará a cabo si el número fue formateado correctamente lo cual se podrá ver en pantalla cuando se realice la llamada.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 7s

Realizado por: Wilmer B. 2018.

Sprint 9

Tabla historia de usuario 16.

HISTORIA DE USUARIO		
Id: HU_16	**Nombre:** Consumo de servicios web a través de petición POST	
Descripción: Como desarrollador necesito implementar una funcionalidad en la cual me permita tener una conexión a internet para el consumo de servicios web del hosting, la misma se realizará con peticiones http por el método post		
Usuario: Wilmer Barrera	**Sprint:** 9	
Fecha inicio: 24/01/18	**Fecha fin:** 24/01/18	**Esfuerzo:** 5

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Implementar funcionalidad de conexión a servicio web alojado en hosting
TI_02	Implementación de funcionalidad para realizar petición a servicio web
TI_03	Creación de interface java para procesar respuesta de servicio web

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 16.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 9
Nombre de historia usuario: HU_16 Consumo de servicios web a través de petición POST		
Nombre tarea: Implementar funcionalidad de conexión a servicio web alojado en hosting.		
Fecha inicio: 24/01/18		**Fecha fin:** 24/01/18
Descripción: Como desarrollador necesito implementar la funcionalidad que me permita conectar con el servicio web alojado en el hosting		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_138	Verificar que la conexión funcione correctamente, conectándose a un servicio web de prueba.	
PA_139	Verificar que dicha conexión permita enviar y recibir información extra hacia y desde el servicio web	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 138.

PRUEBA DE ACEPTACIÓN	
Id: PA_138	**Nombre:** Verificar que la conexión funcione correctamente, conectándose a un servicio web de prueba
Tarea de ingeniería: TI_01 Implementar funcionalidad de conexión a servicio web alojado en hosting	
Descripción: Se debe verificar que la conexión al servicio web mediante el método post sea correcta, ya que este es el método de respuesta del servicio web	
Responsable: Wilmer Barrera	**Fecha:** 24/01/18
Condición de ejecución: - Tener realizada la conexión del servicio web mediante el método post. - Tener desarrollado los servicios web.	

Pasos de ejecución: - Ejecutar el emulador con la aplicación. - Realizar petición con las credenciales para el acceso al servicio web. - Verificar si existe respuestas desde el servicio web.
Resultado esperado: Existe una respuesta satisfactoria desde los servicios web.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 139.

PRUEBA DE ACEPTACIÓN	
Id: PA_139	**Nombre:** Verificar que dicha conexión permita enviar y recibir información extra hacia y desde el servicio web
Tarea de ingeniería: TI_01 Implementar funcionalidad de conexión a servicio web alojado en hosting	
Descripción: desde la conexión hacia el servicio web alojado en el hosting se tiene que enviar parámetros de petición, así como recibir respuesta en formato json	
Responsable: Wilmer Barrera	**Fecha:** 24/01/18
Condición de ejecución: - Tener creado los servicios web - Tener creado los archivos de conexión en la app móvil - Tener agregado credenciales de acceso a servicios web	
Pasos de ejecución: - Verificar la extracción de datos mediante petición post - Verificar que se extraiga credenciales de acceso a servicios web - Verificar que las credenciales de acceso sean válidas	
Resultado esperado: los parámetros de que se envían en una petición mediante el método post son extraídos con éxito, así como las respuestas devueltas por el servicio web se extrae en la aplicación en formato json	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 02 de historia de usuario 16.

TAREA DE INGENIERÍA		
Id: TI_02	**Tipo de tarea:** desarrollo	**Sprint:** 9
Nombre de historia usuario: HU_16 Consumo de servicios web a través de petición POST		
Nombre tarea: Implementación de funcionalidad para realizar petición a servicio web.		
Fecha inicio: 24/01/18		**Fecha fin:** 24/01/18
Descripción: Como desarrollador necesito crear la funcionalidad en la que se realice la petición hacia los servicios web alojados en el hosting, el mismo que será implementado para usar en tal caso.		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_140	Verificar que en la petición en envíe lista de parámetros necesarios para acceder a los servicios web	
PA_141	Verificar que la respuesta a la petición sea a través de una respuesta de petición http post	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 140.

PRUEBA DE ACEPTACIÓN	
Id: PA_140	**Nombre:** Verificar que en la petición en envíe lista de parámetros necesarios para acceder a los servicios web
Tarea de ingeniería: TI_02 Implementación de funcionalidad para realizar petición a servicio web	
Descripción: para poder tener una conexión exitosa con los servicios web es necesario enviar parámetros necesarios, como el servicio web que se va a ejecutar, así como también las credenciales de acceso	
Responsable: Wilmer Barrera	**Fecha:** 24/01/18
Condición de ejecución: - Tener implementado los servicios web - Tener creado los métodos de autenticación - Tener creado la funcionalidad de conexión a los servicios web php desde la app	
Pasos de ejecución: - Verificar que se encuentre definidas las credenciales de autenticación con el servicio web - Verificar que se encuentren definidas las variables de servicio web a ser consumidos	

- Verificar que tanto las credenciales de acceso y variables de acción de servicio web se agreguen automáticamente a una petición desde la app.
Resultado esperado: Los parámetros necesarios para llevar a cabo un consumo de un servicio web específico se agregan automáticamente en una petición desde la app

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 141.

PRUEBA DE ACEPTACIÓN	
Id: PA_141	**Nombre:** Verificar que la respuesta a la petición sea a través de una respuesta de petición http post
Tarea de ingeniería: TI_02 Implementación de funcionalidad para realizar petición a servicio web	
Descripción: cuando se realiza una petición de consumo de servicio web, esta debe traer a través de un resultado por medio del método post.	
Responsable: Wilmer Barrera	**Fecha:** 24/01/18
Condición de ejecución: - Tener creada la funcionalidad para consumir servicios web desde la app móvil - Tener creados los servicios web	
Pasos de ejecución: - Verificar que en las configuraciones de la petición se establezca una entrada de respuesta mediante método post - Verificar que exista respuesta del servicio web	
Resultado esperado: la respuesta del servicio web es mediante el método post, el mismo que devuelve un resultado en una cadena de texto plano en formato json	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 03 de historia de usuario 16.

TAREA DE INGENIERÍA		
Id: TI_03	**Tipo de tarea:** desarrollo	**Sprint:** 9
Nombre de historia usuario: HU_16 Consumo de servicios web a través de petición POST		
Nombre tarea: Creación de interface java para procesar respuesta de servicio web.		
Fecha inicio: 24/01/18		**Fecha fin:** 24/01/18
Descripción:		

Con el fin de extraer resultados o estados de peticiones realizadas a los servicios web y que se encuentran ejecutándose en segundo plano, se implementará la interfaz en la cual permita procesar estos resultados.	
PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_142	Verificar que la interfaz se ejecute cuando se haya traído respuesta desde el servicio web
PA_143	Verificar que la interfaz se ejecute cuando haya pasado el tiempo máximo establecido para obtener una respuesta del servicio web

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 142.

PRUEBA DE ACEPTACIÓN	
Id: PA_142	**Nombre:** Verificar que la interfaz se ejecute cuando se haya traído respuesta desde el servicio web
Tarea de ingeniería: TI_03 Creación de interface java para procesar respuesta de servicio web	
Descripción: la interfaz java debe ejecutarse cuando se haya terminado de traer una respuesta desde el consumo del servicio web, el mismo que se esperará en segundo plano en la app móvil	
Responsable: Wilmer Barrera	**Fecha:** 24/01/18
Condición de ejecución: - Tener implementada la funcionalidad de consumo de servicios web - Tener implementados los servicios web - Tener implementado la funcionalidad para consumo de servicios web en segundo plano	
Pasos de ejecución: - Ejecutar el emulador con la app móvil - Realizar una petición de prueba a los servicios web en el hosting - Imprimir mensaje en la interfaz java cuando se haya obtenido una respuesta o no se haya podido establecer una conexión con los servicios web.	
Resultado esperado: la aplicación muestra un mensaje cuando termina la ejecución de petición de consumo de servicio web en segundo plano, mostrando un mensaje en caso de que traiga o no respuesta del servicio web	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 143.

PRUEBA DE ACEPTACIÓN	
Id: PA_143	**Nombre:** Verificar que la interfaz se ejecute cuando haya pasado el tiempo máximo establecido para obtener una respuesta del servicio web
Tarea de ingeniería: TI_03 Creación de interface java para procesar respuesta de servicio web	
Descripción: cuando se realice una petición de consumo de servicio web este debe definirse un tiempo de espera de respuesta por parte del servicio web. Caso contrario se detendrá la ejecución de petición en segundo plano	
Responsable: Wilmer Barrera	**Fecha:** 24/01/18
Condición de ejecución: - Tener implementado los servicios web - Tener implementado el archivo de conexión hacia los servicios web desde la app móvil	
Pasos de ejecución: - Iniciar el emulador con la última versión de la app - Realizar petición de consumo de servicio web - Verificar que exista respuesta desde el servicio web - Realizar petición a servicio web que no existe - Verificar que pasado el tiempo de respuesta se detenga la ejecución de petición en segundo plano y muestre un mensaje de advertencia, que no se conectó a servicio web	
Resultado esperado: Cuando se consume un servicio web y a conexión es exitosa, entonces se muestra un mensaje donde dice que se trae resultados del servicio web, mientras que si no logró conectar al servicio web y pasó el tiempo estimado de respuesta de esta muestra un mensaje en el que advierte que la conexión al servicio web fue fallida.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla historia de usuario 17.

HISTORIA DE USUARIO		
Id: HU_17	**Nombre:** Sincronización de app móvil con base de datos alojado en servicio de hosting	
Descripción: Como desarrollador necesito poder implementar la funcionalidad en la que me permita sincronizar los datos de las tablas de la base de datos del servicio de hosting con la base de datos local		
Usuario: Wilmer Barrera	**Sprint:** 9	
Fecha inicio: 25/01/18	**Fecha fin:** 01/02/18	**Esfuerzo:** 30
TAREAS DE INGENIERÍA		

Id	Nombre
TI_01	Implementar funcionalidad para verificar que el dispositivo no esté conectado a datos móviles
TI_02	Implementar funcionalidad para realizar petición de consumo de servicio web de hosting para sincronización de base de datos
TI_03	Implementar funcionalidad para extracción de resultados de petición de sincronización.
TI_04	Implementar funcionalidad para la eliminación de contenido y tablas anteriores de la base de datos local
TI_05	Implementar funcionalidad para extracción de tablas desde respuesta de servicio web a la base de datos local.
TI_06	Implementar funcionalidad para inserción de datos de tablas locales con la información extraída desde la respuesta del servicio web
TI_07	Implementar funcionalidad para ejecución de sincronización en segundo plano

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 17.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** desarrollo	**Sprint:** 9
Nombre de historia usuario: HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting		
Nombre tarea: Implementar funcionalidad para verificar que el dispositivo no esté conectado a datos móviles.		
Fecha inicio: 25/01/18		**Fecha fin:**25/01/18
Descripción: Con el fin de ahorrar los paquetes de datos móviles al usuario se implementará un control en el cual advierta al usuario que se consumirán sus datos móviles		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_144	Verificar activación de servicio de wifi	
PA_145	Verificar conexión a la red activa	
PA_146	Verificar cuando no hay conexión a la red activa	
PA_147	Verificar que el servicio de conexión a la red esté disponible	
PA_148	Verificar que el tipo de conexión sea wifi	
PA_149	Verificar que el tipo de conexión de res sea datos móviles	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 144.

PRUEBA DE ACEPTACIÓN	
Id: PA_144	**Nombre:** Verificar activación de servicio de wifi
Tarea de ingeniería: TI_01 Implementar funcionalidad para verificar que el dispositivo no esté conectado a datos móviles.	
Descripción: para poder tener una conexión a la red, esta se debe verificar la activación del servicio wifi	
Responsable: Wilmer Barrera	**Fecha:** 25/01/18
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desactivado la conexión wifi - Tener desarrollada la funcionalidad para activar el servicio de conexión a través de wifi	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar botón para activar la conexión a través de wifi - Verificar el tipo de respuesta que se muestra en la consola	
Resultado esperado: cuando se verifica que la conexión a través de wifi se haya activado, este proceso debe activar automáticamente la conexión a través de wifi en el emulador e imprimir la respuesta booleana que arroja este proceso, esto confirma que se ha activado dicha conexión	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 145.

PRUEBA DE ACEPTACIÓN	
Id: PA_145	**Nombre:** Verificar conexión a la red activa
Tarea de ingeniería: TI_01 Implementar funcionalidad para verificar que el dispositivo no esté conectado a datos móviles.	
Descripción: cuando se verifica la conexión a una red activa, el sistema deberá verificar esta situación y actuar ante esta.	
Responsable: Wilmer Barrera	**Fecha:** 25/01/18
Condición de ejecución: - Tener activado datos móviles o - Tener activado la conexión por wifi - Tener instalada la última versión de la app en el emulador	

- Tener desarrollada la funcionalidad para verificar la existencia de conexión a una red activa
Pasos de ejecución: - Ejecutar la app desde el emulador - Presiona el botón para verificar si existe conexión a una red activa - Verificar el resultado presentado en consola
Resultado esperado: cuando se mande a verificar que existe una conexión a una red activa, este proceso devolverá un valor booleano igual a false, lo que significa que existe una conexión a una red activa.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 146.

PRUEBA DE ACEPTACIÓN	
Id: PA_146	**Nombre:** Verificar cuando no hay conexión a la red activa
Tarea de ingeniería: TI_01 Implementar funcionalidad para verificar que el dispositivo no esté conectado a datos móviles.	
Descripción: cuando se verifica la conexión a una red activa y esta no existe, el sistema deberá verificar esta situación y actuar ante esta.	
Responsable: Wilmer Barrera	**Fecha:** 25/01/18
Condición de ejecución: - Tener desactivado datos móviles - Tener desactivado la conexión por wifi - Tener instalada la última versión de la app en el emulador - Tener desarrollada la funcionalidad para verificar la existencia de conexión a una red activa	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presiona el botón para verificar si existe conexión a una red activa - Verificar el resultado presentado en consola	
Resultado esperado: cuando se mande a verificar que existe una conexión a una red activa y esta no existe, este proceso devolverá un valor booleano igual a true, lo que significa que no existe una conexión a una red activa.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 147.

PRUEBA DE ACEPTACIÓN	
Id: PA_147	**Nombre:** Verificar que el servicio de conexión a la red esté disponible
Tarea de ingeniería: TI_01 Implementar funcionalidad para verificar que el dispositivo no esté conectado a datos móviles.	
Descripción: para poder tener una conexión exitosa a la red se debe verificar antes si la red si la conexión a la red está disponible.	
Responsable: Wilmer Barrera	**Fecha:** 25/01/18
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener creada la funcionalidad para verificar la disponibilidad de la red - Tener activada la conexión a la red a través de datos móviles o red wifi	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar el botón para verificar conexión a la red - Verificar la respuesta imprimida en el emulador.	
Resultado esperado: cuando se verifica la disponibilidad de la red, este proceso devolverá un valor booleano igual a true, lo cual significa que existe la red disponible.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 148.

PRUEBA DE ACEPTACIÓN	
Id: PA_148	**Nombre:** Verificar que el tipo de conexión sea wifi
Tarea de ingeniería: TI_01 Implementar funcionalidad para verificar que el dispositivo no esté conectado a datos móviles.	
Descripción: cuando se requiera verificar la conexión de red para realizar la sincronización de datos, se debe verificar que la conexión sea a través de red wifi	
Responsable: Wilmer Barrera	**Fecha:** 25/01/18
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desarrollada la funcionalidad para verificar cual es el proveedor de internet - Tener activada la conexión wifi	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar botón para verificar que la conexión a la red sea a través de wifi	

- Verificar la respuesta imprimida en la consola
Resultado esperado: cuando se verifica que la conexión a la red sea mediante wifi, este proceso devolverá como resultado una cadena de texto igual a “wifi” lo que significa que si está conectado a la red mediante wifi.
Evaluación de la prueba: Exitosa | **Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 149.

PRUEBA DE ACEPTACIÓN	
Id: PA_149	**Nombre:** Verificar que el tipo de conexión de res sea datos móviles
Tarea de ingeniería: TI_01 Implementar funcionalidad para verificar que el dispositivo no esté conectado a datos móviles.	
Descripción: cuando se requiera verificar la conexión de red para realizar la sincronización de datos, se debe verificar que la conexión sea a través de red de datos móviles.	
Responsable: Wilmer Barrera	**Fecha:** 25/01/18
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener desarrollada la funcionalidad para verificar cual es el proveedor de internet - Tener activada la conexión a través de datos móviles	
Pasos de ejecución: - Ejecutar la app desde el emulador - Presionar botón para verificar el tipo de conexión a la red - Verificar la respuesta imprimida en la consola	
Resultado esperado: cuando se verifica que la conexión a la red sea mediante datos móviles, este proceso devolverá como resultado una cadena de texto igual a “mobile” lo que significa que si está conectado a la red mediante datos móviles.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 02 de historia de usuario 17.

TAREA DE INGENIERÍA		
Id: TI_02	**Tipo de tarea:** desarrollo	**Sprint:** 9
Nombre de historia usuario: HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting		
Nombre tarea: Implementar funcionalidad para realizar petición de consumo de servicio web de hosting para sincronización de base de datos.		

Fecha inicio: 26/01/18	**Fecha fin:**26/01/18
Descripción: Con el fin de realizar consumo de servicios web específicamente al momento de sincronizar las bases de datos se implementará la funcionalidad necesaria para llevar a cabo dicha petición	
PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_150	Verificar que la petición http sea aceptada.

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 150.

PRUEBA DE ACEPTACIÓN	
Id: PA_150	**Nombre:** Verificar que la petición http sea aceptada.
Tarea de ingeniería: TI_02 Implementar funcionalidad para realizar petición de consumo de servicio web de hosting para sincronización de base de datos.	
Descripción: cuando se realice la petición para sincronizar los datos, esta petición se debe verificar que sea aceptada en caso de estar bien estructurada.	
Responsable: Wilmer Barrera	**Fecha:** 26/01/18
Condición de ejecución: - Tener instalada la última versión de la app móvil en el emulador. - Tener una conexión activa a internet - Tener desarrollada la funcionalidad de conexión a servicios web	
Pasos de ejecución: - Ingresar el parámetro de la petición, este parámetro es la url del servicio web - Ejecutar la app desde el emulador - Verificar la respuesta que se muestra en la consola.	
Resultado esperado: cuando se realiza la petición de sincronización, si este proceso se lleva a cabo correctamente devolverá un resultado booleano igual a true, lo que significa que la petición fue aceptada.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 03 de historia de usuario 17.

TAREA DE INGENIERÍA		
Id: TI_03	**Tipo de tarea:** desarrollo	**Sprint:** 9
Nombre de historia usuario: HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting		

Nombre tarea: Implementar funcionalidad para extracción de resultados de petición de sincronización.	
Fecha inicio: 29/01/18	**Fecha fin:**29/01/18
Descripción: Al momento de haber realizado una petición de sincronización a los servicios web de hosting este proceso quedará aun ejecutándose en segundo plano esperando una respuesta desde el servidor, la misma que cuando ya traiga una respuesta esta será procesada para extraer los datos necesarios para completar la sincronización	

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_151	Verificar que la respuesta del servicio web haya sido exitosa
PA_152	Verificar que la respuesta del servicio web haya sido fallida

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 151.

PRUEBA DE ACEPTACIÓN	
Id: PA_151	**Nombre:** Verificar que la respuesta del servicio web haya sido exitosa
Tarea de ingeniería: TI_03 Implementar funcionalidad para extracción de resultados de petición de sincronización.	
Descripción: cuando se realiza la petición de sincronización de datos, se debe verificar si existe una respuesta de regreso positivo antes esta petición.	
Responsable: Wilmer Barrera	**Fecha:** 29/01/18
Condición de ejecución: - Tener desarrollada la conexión hacia los servicios web - Tener instalada la última versión de la app móvil en el emulador - Tener una conexión activa a internet	
Pasos de ejecución: - Ingresar el parámetro de la petición para sincronizar la app móvil, este parámetro es el url del servicio web - Ejecutar la app desde el emulador - Verificar la respuesta extraída del servicio web.	
Resultado esperado: cuando se extraiga el resultado del servicio web de este se extraerá el estado de la respuesta que es una cadena de texto igual a “CC” lo que significa que exista una respuesta con datos.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 5s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 152.

PRUEBA DE ACEPTACIÓN	
Id: PA_152	**Nombre:** Verificar que la respuesta del servicio web haya sido fallida
Tarea de ingeniería: TI_03 Implementar funcionalidad para extracción de resultados de petición de sincronización.	
Descripción: cuando se realiza la petición de sincronización de datos, se debe verificar cuando no existen datos en la respuesta del servicio web.	
Responsable: Wilmer Barrera	**Fecha:** 29/01/18
Condición de ejecución: - Tener desarrollada la conexión hacia los servicios web - Tener instalada la última versión de la app móvil en el emulador - Tener una conexión activa a internet	
Pasos de ejecución: - Ingresar el parámetro de la petición para sincronizar la app móvil, este parámetro es el url del servicio web - Ejecutar la app desde el emulador - Verificar la respuesta extraída del servicio web.	
Resultado esperado: cuando se extraiga el resultado del servicio web de este se extraerá el estado de la respuesta que es una cadena de texto igual a "CI" lo que significa que exista una respuesta sin datos.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 04 de historia de usuario 17.

TAREA DE INGENIERÍA		
Id: TI_04	**Tipo de tarea:** desarrollo	**Sprint:** 9
Nombre de historia usuario: HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting		
Nombre tarea: Implementar funcionalidad para la eliminación de contenido y tablas anteriores de la base de datos local.		
Fecha inicio: 30/01/18		**Fecha fin:**30/01/18
Descripción: Al momento de sincronizar la aplicación móvil, se necesita que se implemente una funcionalidad en la que elimine el contenido de las tablas anteriores para actualizar el contenido de estas.		
PRUEBAS DE ACEPTACIÓN		

Id	Nombre
PA_153	Verificar la existencia de las tablas: COOPERATIVA, DEPENDENCIA, TERMINAL, RUTA, HORARIO, TERM_PERTENECE_DEPEN, DEPEN_POSEE_RUTAS, PROVINCIAS, CIUDADES, FAVORITO de la base de datos local.
PA_154	Verificar la eliminación de contenido de todas las tablas de la base de datos local.

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 153.

PRUEBA DE ACEPTACIÓN	
Id: PA_153	**Nombre:** Verificar la existencia de las tablas: COOPERATIVA, DEPENDENCIA, TERMINAL, RUTA, HORARIO, TERM_PERTENECE_DEPEN, DEPEN_POSEE_RUTAS, PROVINCIAS, CIUDADES, FAVORITO de la base de datos local.
Tarea de ingeniería: TI_04 Implementar funcionalidad para la eliminación de contenido y tablas anteriores de la base de datos local.	
Descripción: para poder sincronizar los datos de la base de datos local se debe verificar que todas las tablas de esta tengan datos con el fin de ejecutar un proceso de eliminación o no.	
Responsable: Wilmer Barrera	**Fecha:** 30/01/18
Condición de ejecución: - Tener instalada la última versión de la app en el emulador - Tener creada la base de datos - Tener creada las tablas: COOPERATIVA, DEPENDENCIA, TERMINAL, RUTA, HORARIO, TERM_PERTENECE_DEPEN, DEPEN_POSEE_RUTAS, PROVINCIAS, CIUDADES, FAVORITO, cada una de estas con datos. - Tener creada la conexión a la base de datos - Tener una conexión activa a internet	
Pasos de ejecución: - Ingresar el parámetro para verificar el contenido de la tabla, este parámetro es el nombre de las tablas: COOPERATIVA, DEPENDENCIA, TERMINAL, RUTA, HORARIO, TERM_PERTENECE_DEPEN, DEPEN_POSEE_RUTAS, PROVINCIAS, CIUDADES, FAVORITO, en cada una de las pruebas manuales correspondientes. - Ejecutar la app móvil desde el emulador - Verificar el resultado mostrado que se muestra en consola.	

Resultado esperado: cuando se verifique el contenido de datos en cada una de las tablas en cada prueba manual este proceso deberá regresar un resultado entero mayor a cero, el cual corresponde al número de registros almacenados en dicha tabla.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 10 x 3 = 30s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 154.

PRUEBA DE ACEPTACIÓN	
Id: PA_154	**Nombre:** Verificar la eliminación de contenido de todas las tablas de la base de datos local.
Tarea de ingeniería: TI_04 Implementar funcionalidad para la eliminación de contenido y tablas anteriores de la base de datos local.	
Descripción: cuando no se necesita verificar si una tabla en la base de datos local tiene o no contenido esta tabla, se puede mandar a eliminar el contenido de todas las tablas, y se debe verificar que esta eliminación se haya llevado a cabo correctamente.	
Responsable: Wilmer Barrera	**Fecha:** 30/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creadas las tablas: COOPERATIVA, DEPENDENCIA, TERMINAL, RUTA, HORARIO, TERM_PERTENECE_DEPEN, DEPEN_POSEE_RUTAS, PROVINCIAS, CIUDADES, FAVORITO. - Tener creada la conexión a la base de datos - Tener instalada la última versión de la app móvil.	
Pasos de ejecución: - Ejecutar la app móvil desde el emulador - Presionar el botón para eliminar el contenido de todas las tablas - Verificar la respuesta que se muestra en la consola de la aplicación.	
Resultado esperado: si la eliminación de todos los datos de cada una de las tablas se llevó a cabo correctamente, este proceso devolverá un valor booleano verdadero que se imprimirá en pantalla.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 05 de historia de usuario 17.

TAREA DE INGENIERÍA		
Id: TI_05	**Tipo de tarea:** desarrollo	**Sprint:** 9
Nombre de historia usuario: HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting		
Nombre tarea: Implementar funcionalidad para extracción de tablas desde respuesta de servicio web a la base de datos local.		
Fecha inicio: 30/01/18		**Fecha fin:**30/01/18
Descripción: Como desarrollador necesito crear la funcionalidad en la que se extraerán cada una de las tablas que posteriormente serán insertadas en las tablas de la base de datos local.		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_155	Verificar que la tabla COOPERATIVA sea extraída de la respuesta del servicio web	
PA_156	Verificar que la tabla DEPENDENCIA sea extraída de la respuesta del servicio web	
PA_157	Verificar que la tabla TERMINAL sea extraída de la respuesta del servicio web	
PA_158	Verificar que la tabla RUTA sea extraída de la respuesta del servicio web	
PA_159	Verificar que la tabla HORARIO sea extraída de la respuesta del servicio web	
PA_160	Verificar que la tabla TERM_PERTENECE_DEPEN sea extraída de la respuesta del servicio web	
PA_161	Verificar que la tabla DEPEN_POSEE_RUTAS sea extraída de la respuesta del servicio web	
PA_162	Verificar que la tabla PROVINCIAS sea extraída de la respuesta del servicio web	
PA_163	Verificar que la tabla CIUDADES sea extraída de la respuesta del servicio web	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 155.

PRUEBA DE ACEPTACIÓN	
Id: PA_155	**Nombre:** Verificar que la tabla COOPERATIVA sea extraída de la respuesta del servicio web
Tarea de ingeniería: TI_05 Implementar funcionalidad para extracción de tablas desde respuesta de servicio web a la base de datos local.	
Descripción: cuando se lleva a cabo el proceso de sincronización de las tablas se debe verificar que cada una de las tablas sean extraídas, en este caso se verificará que la tabla COOPERATIVA	
Responsable: Wilmer Barrera	**Fecha:** 30/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla COOPERATIVA - Tener creada la función para extraer datos de tablas en sincronización - Tener instalada la última versión de la app móvil - Tener una conexión activa a internet	
Pasos de ejecución: - Insertar el parámetro de extracción de datos, este parámetro es el nombre de la tabla a extraer sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.	
Resultado esperado: al mandar a verificar que la extracción de datos de la tabla COOPERATIVA se ha llevado con éxito, este proceso devolverá un resultado booleano igual a true, que confirma que la extracción fue exitosa.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 156.

PRUEBA DE ACEPTACIÓN	
Id: PA_156	**Nombre:** Verificar que la tabla DEPENDENCIA sea extraída de la respuesta del servicio web
Tarea de ingeniería: TI_05 Implementar funcionalidad para extracción de tablas desde respuesta de servicio web a la base de datos local.	

Descripción: cuando se lleva a cabo el proceso de sincronización de las tablas se debe verificar que cada una de las tablas sean extraídas, en este caso se verificará que la tabla DEPENDENCIA	
Responsable: Wilmer Barrera	**Fecha:** 30/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla DEPENDENCIA - Tener creada la función para extraer datos de tablas en sincronización - Tener instalada la última versión de la app móvil - Tener una conexión activa a internet	
Pasos de ejecución: - Insertar el parámetro de extracción de datos, este parámetro es el nombre de la tabla a extraer sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.	
Resultado esperado: al mandar a verificar que la extracción de datos de la tabla DEPENDENCIA se ha llevado con éxito, este proceso devolverá un resultado booleano igual a true, que confirma que la extracción fue exitosa.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 157.

PRUEBA DE ACEPTACIÓN	
Id: PA_157	**Nombre:** Verificar que la tabla TERMINAL sea extraída de la respuesta del servicio web
Tarea de ingeniería: TI_05 Implementar funcionalidad para extracción de tablas desde respuesta de servicio web a la base de datos local.	
Descripción: cuando se lleva a cabo el proceso de sincronización de las tablas se debe verificar que cada una de las tablas sean extraídas, en este caso se verificará que la tabla TERMINAL	
Responsable: Wilmer Barrera	**Fecha:** 30/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla TERMINAL - Tener creada la función para extraer datos de tablas en sincronización - Tener instalada la última versión de la app móvil	

- Tener una conexión activa a internet
Pasos de ejecución: - Insertar el parámetro de extracción de datos, este parámetro es el nombre de la tabla a extraer sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.
Resultado esperado: al mandar a verificar que la extracción de datos de la tabla TERMINAL se ha llevado con éxito, este proceso devolverá un resultado booleano igual a true, que confirma que la extracción fue exitosa.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 158.

PRUEBA DE ACEPTACIÓN	
Id: PA_158	**Nombre:** Verificar que la tabla RUTA sea extraída de la respuesta del servicio web
Tarea de ingeniería: TI_05 Implementar funcionalidad para extracción de tablas desde respuesta de servicio web a la base de datos local.	
Descripción: cuando se lleva a cabo el proceso de sincronización de las tablas se debe verificar que cada una de las tablas sean extraídas, en este caso se verificará que la tabla RUTA	
Responsable: Wilmer Barrera	**Fecha:** 30/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla RUTA - Tener creada la función para extraer datos de tablas en sincronización - Tener instalada la última versión de la app móvil - Tener una conexión activa a internet	
Pasos de ejecución: - Insertar el parámetro de extracción de datos, este parámetro es el nombre de la tabla a extraer sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.	

Resultado esperado: al mandar a verificar que la extracción de datos de la tabla RUTA se ha llevado con éxito, este proceso devolverá un resultado booleano igual a true, que confirma que la extracción fue exitosa.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 159.

PRUEBA DE ACEPTACIÓN	
Id: PA_159	**Nombre:** Verificar que la tabla HORARIO sea extraída de la respuesta del servicio web
Tarea de ingeniería: TI_05 Implementar funcionalidad para extracción de tablas desde respuesta de servicio web a la base de datos local.	
Descripción: cuando se lleva a cabo el proceso de sincronización de las tablas se debe verificar que cada una de las tablas sean extraídas, en este caso se verificará que la tabla HORARIO	
Responsable: Wilmer Barrera	**Fecha:** 30/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla HORARIO - Tener creada la función para extraer datos de tablas en sincronización - Tener instalada la última versión de la app móvil - Tener una conexión activa a internet	
Pasos de ejecución: - Insertar el parámetro de extracción de datos, este parámetro es el nombre de la tabla a extraer sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.	
Resultado esperado: al mandar a verificar que la extracción de datos de la tabla HORARIO se ha llevado con éxito, este proceso devolverá un resultado booleano igual a true, que confirma que la extracción fue exitosa.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 160.

PRUEBA DE ACEPTACIÓN	
Id: PA_160	**Nombre:** Verificar que la tabla TERM_PERTENECE_DEPEN sea extraída de la respuesta del servicio web
Tarea de ingeniería: TI_05 Implementar funcionalidad para extracción de tablas desde respuesta de servicio web a la base de datos local.	
Descripción: cuando se lleva a cabo el proceso de sincronización de las tablas se debe verificar que cada una de las tablas sean extraídas, en este caso se verificará que la tabla TERM_PERTENECE_DEPEN	
Responsable: Wilmer Barrera	**Fecha:** 30/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla TERM_PERTENECE_DEPEN - Tener creada la función para extraer datos de tablas en sincronización - Tener instalada la última versión de la app móvil - Tener una conexión activa a internet	
Pasos de ejecución: - Insertar el parámetro de extracción de datos, este parámetro es el nombre de la tabla a extraer sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.	
Resultado esperado: al mandar a verificar que la extracción de datos de la tabla TERM_PERTENECE_DEPEN se ha llevado con éxito, este proceso devolverá un resultado booleano igual a true, que confirma que la extracción fue exitosa.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 161.

PRUEBA DE ACEPTACIÓN	
Id: PA_161	**Nombre:** Verificar que la tabla DEPEN_POSEE_RUTAS sea extraída de la respuesta del servicio web
Tarea de ingeniería: TI_05 Implementar funcionalidad para extracción de tablas desde respuesta de servicio web a la base de datos local.	

Descripción: cuando se lleva a cabo el proceso de sincronización de las tablas se debe verificar que cada una de las tablas sean extraídas, en este caso se verificará que la tabla DEPEN_POSEE_RUTAS	
Responsable: Wilmer Barrera	**Fecha:** 30/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla DEPEN_POSEE_RUTAS - Tener creada la función para extraer datos de tablas en sincronización - Tener instalada la última versión de la app móvil - Tener una conexión activa a internet	
Pasos de ejecución: - Insertar el parámetro de extracción de datos, este parámetro es el nombre de la tabla a extraer sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.	
Resultado esperado: al mandar a verificar que la extracción de datos de la tabla DEPEN_POSEE_RUTAS se ha llevado con éxito, este proceso devolverá un resultado booleano igual a true, que confirma que la extracción fue exitosa.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 162.

PRUEBA DE ACEPTACIÓN	
Id: PA_162	**Nombre:** Verificar que la tabla PROVINCIAS sea extraída de la respuesta del servicio web
Tarea de ingeniería: TI_05 Implementar funcionalidad para extracción de tablas desde respuesta de servicio web a la base de datos local.	
Descripción: cuando se lleva a cabo el proceso de sincronización de las tablas se debe verificar que cada una de las tablas sean extraídas, en este caso se verificará que la tabla PROVINCIAS	
Responsable: Wilmer Barrera	**Fecha:** 30/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla PROVINCIAS - Tener creada la función para extraer datos de tablas en sincronización - Tener instalada la última versión de la app móvil	

- Tener una conexión activa a internet
Pasos de ejecución: - Insertar el parámetro de extracción de datos, este parámetro es el nombre de la tabla a extraer sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.
Resultado esperado: al mandar a verificar que la extracción de datos de la tabla PROVINCIAS se ha llevado con éxito, este proceso devolverá un resultado booleano igual a true, que confirma que la extracción fue exitosa.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 163.

PRUEBA DE ACEPTACIÓN	
Id: PA_163	**Nombre:** Verificar que la tabla CIUDADES sea extraída de la respuesta del servicio web
Tarea de ingeniería: TI_05 Implementar funcionalidad para extracción de tablas desde respuesta de servicio web a la base de datos local.	
Descripción: cuando se lleva a cabo el proceso de sincronización de las tablas se debe verificar que cada una de las tablas sean extraídas, en este caso se verificará que la tabla CIUDADES	
Responsable: Wilmer Barrera	**Fecha:** 30/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla CIUDADES - Tener creada la función para extraer datos de tablas en sincronización - Tener instalada la última versión de la app móvil - Tener una conexión activa a internet	
Pasos de ejecución: - Insertar el parámetro de extracción de datos, este parámetro es el nombre de la tabla a extraer sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.	

Resultado esperado: al mandar a verificar que la extracción de datos de la tabla CIUDADES se ha llevado con éxito, este proceso devolverá un resultado booleano igual a true, que confirma que la extracción fue exitosa.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 06 de historia de usuario 17.

TAREA DE INGENIERÍA		
Id: TI_06	**Tipo de tarea:** desarrollo	**Sprint:** 9
Nombre de historia usuario: HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting		
Nombre tarea: Implementar funcionalidad para inserción de datos de tablas locales con la información extraída desde la respuesta del servicio web.		
Fecha inicio: 31/01/18		**Fecha fin:** 31/01/18
Descripción: Como desarrollador necesito crear la funcionalidad en la que se realice la inserción de todas las tablas extraídas como resultado de la petición de sincronización y se inserten en las tablas de base de datos local.		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_164	Verificar la inserción correcta de los datos de la tabla COOPERATIVA
PA_165	Verificar la inserción correcta de los datos de la tabla DEPENDENCIA
PA_166	Verificar la inserción correcta de los datos de la tabla TERMINAL
PA_167	Verificar la inserción correcta de los datos de la tabla RUTA
PA_168	Verificar la inserción correcta de los datos de la tabla HORARIO
PA_169	Verificar la inserción correcta de los datos de la tabla TERM_PERTENECE_DEPEN
PA_170	Verificar la inserción correcta de los datos de la tabla DEPEN_POSEE_RUTAS
PA_171	Verificar la inserción correcta de los datos de la tabla PROVINCIAS
PA_172	Verificar la inserción correcta de los datos de la tabla CIUDADES

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 164.

PRUEBA DE ACEPTACIÓN	
Id: PA_164	**Nombre:** Verificar la inserción correcta de los datos de la tabla COOPERATIVA
Tarea de ingeniería: TI_06 Implementar funcionalidad para inserción de datos de tablas locales con la información extraída desde la respuesta del servicio web.	
Descripción: cuando se ha extraído los datos de la tabla COOPERATIVA es necesario verificar que estos datos hayan sido insertados en la tabla con el mismo nombre que se encuentra alojada localmente.	
Responsable: Wilmer Barrera	**Fecha:** 31/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla COOPERATIVA - Tener la tabla COOPERATIVA vacía o sin datos registrados - Tener creada la función para insertar datos de tablas en sincronización - Tener instalada la última versión de la app móvil - Tener una conexión activa a internet	
Pasos de ejecución: - Insertar el parámetro de inserción de datos, este parámetro es el nombre de la tabla a la que se insertarán sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.	
Resultado esperado: cuando se envíe a insertar los datos extraídos de la tabla COOPERATIVA, este proceso devolverá un valor booleano igual a true, lo cual significa que la inserción se realizó correctamente.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 165.

PRUEBA DE ACEPTACIÓN	
Id: PA_165	**Nombre:** Verificar la inserción correcta de los datos de la tabla DEPENDENCIA
Tarea de ingeniería: TI_06 Implementar funcionalidad para inserción de datos de tablas locales con la información extraída desde la respuesta del servicio web.	
Descripción: cuando se lleva a cabo el proceso de sincronización de las tablas se debe verificar que cada una de las tablas sean extraídas, en este caso se verificará que la tabla DEPENDENCIA	
Responsable: Wilmer Barrera	**Fecha:** 31/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla DEPENDENCIA - Tener la tabla DEPENDENCIA vacía o sin datos registrados - Tener creada la función para insertar datos de tablas en sincronización - Tener instalada la última versión de la app móvil - Tener una conexión activa a internet	
Pasos de ejecución: - Insertar el parámetro de inserción de datos, este parámetro es el nombre de la tabla a la que se insertarán sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.	
Resultado esperado: cuando se envíe a insertar los datos extraídos de la tabla DEPENDENCIA, este proceso devolverá un valor booleano igual a true, lo cual significa que la inserción se realizó correctamente.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 166.

PRUEBA DE ACEPTACIÓN	
Id: PA_166	**Nombre:** Verificar la inserción correcta de los datos de la tabla TERMINAL
Tarea de ingeniería: TI_06 Implementar funcionalidad para inserción de datos de tablas locales con la información extraída desde la respuesta del servicio web.	
Descripción: cuando se lleva a cabo el proceso de sincronización de las tablas se debe verificar que cada una de las tablas sean extraídas, en este caso se verificará que la tabla TERMINAL	

Responsable: Wilmer Barrera	**Fecha:** 31/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla TERMINAL - Tener la tabla TERMINAL vacía o sin datos registrados - Tener creada la función para insertar datos de tablas en sincronización - Tener instalada la última versión de la app móvil - Tener una conexión activa a internet	
Pasos de ejecución: - Insertar el parámetro de inserción de datos, este parámetro es el nombre de la tabla a la que se insertarán sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.	
Resultado esperado: cuando se envíe a insertar los datos extraídos de la tabla TERMINAL, este proceso devolverá un valor booleano igual a true, lo cual significa que la inserción se realizó correctamente.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 167.

PRUEBA DE ACEPTACIÓN	
Id: PA_167	**Nombre:** Verificar la inserción correcta de los datos de la tabla RUTA
Tarea de ingeniería: TI_06 Implementar funcionalidad para inserción de datos de tablas locales con la información extraída desde la respuesta del servicio web.	
Descripción: cuando se lleva a cabo el proceso de sincronización de las tablas se debe verificar que cada una de las tablas sean extraídas, en este caso se verificará que la tabla RUTA	
Responsable: Wilmer Barrera	**Fecha:** 31/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla RUTA - Tener la tabla RUTA vacía o sin datos registrados - Tener creada la función para insertar datos de tablas en sincronización - Tener instalada la última versión de la app móvil - Tener una conexión activa a internet	
Pasos de ejecución:	

- Insertar el parámetro de inserción de datos, este parámetro es el nombre de la tabla a la que se insertarán sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.
Resultado esperado: cuando se envíe a insertar los datos extraídos de la tabla RUTA, este proceso devolverá un valor booleano igual a true, lo cual significa que la inserción se realizó correctamente.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 7s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 168.

PRUEBA DE ACEPTACIÓN	
Id: PA_168	**Nombre:** Verificar la inserción correcta de los datos de la tabla HORARIO
Tarea de ingeniería: TI_06 Implementar funcionalidad para inserción de datos de tablas locales con la información extraída desde la respuesta del servicio web.	
Descripción: cuando se lleva a cabo el proceso de sincronización de las tablas se debe verificar que cada una de las tablas sean extraídas, en este caso se verificará que la tabla HORARIO	
Responsable: Wilmer Barrera	**Fecha:** 31/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla HORARIO - Tener la tabla HORARIO sin datos registrados - Tener creada la función para insertar datos de tablas en sincronización - Tener instalada la última versión de la app móvil - Tener una conexión activa a internet	
Pasos de ejecución: - Insertar el parámetro de inserción de datos, este parámetro es el nombre de la tabla a la que se insertarán sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.	
Resultado esperado: cuando se envíe a insertar los datos extraídos de la tabla HORARIO, este proceso devolverá un valor booleano igual a true, lo cual significa que la inserción se realizó correctamente.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 8s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 169.

PRUEBA DE ACEPTACIÓN	
Id: PA_169	**Nombre:** Verificar la inserción correcta de los datos de la tabla TERM_PERTENECE_DEPEN
Tarea de ingeniería: TI_06 Implementar funcionalidad para inserción de datos de tablas locales con la información extraída desde la respuesta del servicio web.	
Descripción: cuando se lleva a cabo el proceso de sincronización de las tablas se debe verificar que cada una de las tablas sean extraídas, en este caso se verificará que la tabla TERM_PERTENECE_DEPEN	
Responsable: Wilmer Barrera	**Fecha:** 31/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla TERM_PERTENECE_DEPEN - Tener la tabla TERM_PERTENECE_DEPEN vacía o sin datos registrados - Tener creada la función para insertar datos de tablas en sincronización - Tener instalada la última versión de la app móvil - Tener una conexión activa a internet	
Pasos de ejecución: - Insertar el parámetro de inserción de datos, este parámetro es el nombre de la tabla a la que se insertarán sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.	
Resultado esperado: cuando se envíe a insertar los datos extraídos de la tabla TERM_PERTENECE_DEPEN, este proceso devolverá un valor booleano igual a true, lo cual significa que la inserción se realizó correctamente.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 170.

PRUEBA DE ACEPTACIÓN	
Id: PA_170	**Nombre:** Verificar la inserción correcta de los datos de la tabla DEPEN_POSEE_RUTAS
Tarea de ingeniería: TI_06 Implementar funcionalidad para inserción de datos de tablas locales con la información extraída desde la respuesta del servicio web.	
Descripción: cuando se lleva a cabo el proceso de sincronización de las tablas se debe verificar que cada una de las tablas sean extraídas, en este caso se verificará que la tabla DEPEN_POSEE_RUTAS	
Responsable: Wilmer Barrera	**Fecha:** 31/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla DEPEN_POSEE_RUTAS - Tener la tabla DEPEN_POSEE_RUTAS vacía o sin datos registrados - Tener creada la función para insertar datos de tablas en sincronización - Tener instalada la última versión de la app móvil - Tener una conexión activa a internet	
Pasos de ejecución: - Insertar el parámetro de inserción de datos, este parámetro es el nombre de la tabla a la que se insertarán sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.	
Resultado esperado: cuando se envíe a insertar los datos extraídos de la tabla DEPEN_POSEE_RUTAS, este proceso devolverá un valor booleano igual a true, lo cual significa que la inserción se realizó correctamente.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 4s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 171.

PRUEBA DE ACEPTACIÓN	
Id: PA_171	**Nombre:** Verificar la inserción correcta de los datos de la tabla PROVINCIAS
Tarea de ingeniería: TI_06 Implementar funcionalidad para inserción de datos de tablas locales con la información extraída desde la respuesta del servicio web.	
Descripción: cuando se lleva a cabo el proceso de sincronización de las tablas se debe verificar que cada una de las tablas sean extraídas, en este caso se verificará que la tabla PROVINCIAS	

Responsable: Wilmer Barrera	**Fecha:** 31/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla PROVINCIAS - Tener la tabla PROVINCIAS vacía o sin datos registrados - Tener creada la función para insertar datos de tablas en sincronización - Tener instalada la última versión de la app móvil - Tener una conexión activa a internet	
Pasos de ejecución: - Insertar el parámetro de inserción de datos, este parámetro es el nombre de la tabla a la que se insertarán sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.	
Resultado esperado: cuando se envíe a insertar los datos extraídos de la tabla PROVINCIAS, este proceso devolverá un valor booleano igual a true, lo cual significa que la inserción se realizó correctamente.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 172.

PRUEBA DE ACEPTACIÓN	
Id: PA_172	**Nombre:** Verificar la inserción correcta de los datos de la tabla CIUDADES
Tarea de ingeniería: TI_06 Implementar funcionalidad para inserción de datos de tablas locales con la información extraída desde la respuesta del servicio web.	
Descripción: cuando se lleva a cabo el proceso de sincronización de las tablas se debe verificar que cada una de las tablas sean extraídas, en este caso se verificará que la tabla CIUDADES	
Responsable: Wilmer Barrera	**Fecha:** 31/01/18
Condición de ejecución: - Tener creada la base de datos - Tener creada la tabla CIUDADES - Tener la tabla CIUDADES vacía o sin datos registrados - Tener creada la función para insertar datos de tablas en sincronización - Tener instalada la última versión de la app móvil - Tener una conexión activa a internet	
Pasos de ejecución:	

- Insertar el parámetro de inserción de datos, este parámetro es el nombre de la tabla a la que se insertarán sus datos. - Ejecutar la app móvil desde el emulador - Presionar el botón se sincronización de datos - Verificar la respuesta que devuelve este proceso a través de la consola.
Resultado esperado: cuando se envíe a insertar los datos extraídos de la tabla CIUDADES, este proceso devolverá un valor booleano igual a true, lo cual significa que la inserción se realizó correctamente.

Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 3s

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 07 de historia de usuario 17.

TAREA DE INGENIERÍA		
Id: TI_07	**Tipo de tarea:** desarrollo	**Sprint:** 9
Nombre de historia usuario: HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting		
Nombre tarea: Implementar funcionalidad para ejecución de sincronización en segundo plano.		
Fecha inicio: 01/02/18		**Fecha fin:**01/02/18
Descripción: Con el fin de que la aplicación no entre a un estado de no responder, las peticiones de sincronización se ejecutarán en segundo plano		

PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_173	Verificar que el proceso de sincronización de base de datos se lleve a cabo en segundo plano.

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 173.

PRUEBA DE ACEPTACIÓN	
Id: PA_173	**Nombre:** Verificar que el proceso de sincronización de base de datos se lleve a cabo en segundo plano.
Tarea de ingeniería: TI_07 Implementar funcionalidad para ejecución de sincronización en segundo plano	
Descripción: Para el consumo del servicio web para sincronizar la app es necesario que la petición se ejecute en segundo plano, esto con el fin de que el sistema operativo no asuma que la aplicación se detuvo esperando una respuesta ejecutada en el hilo principal de la app	

Responsable: Wilmer Barrera	**Fecha:** 01/02/18
Condición de ejecución: - Tener creada la conexión al servicio web para la sincronización - Tener instalada la última versión de la app en el emulador	
Pasos de ejecución: - Ejecutar el emulador con la aplicación - Esperar 6 segundos para verificar que la aplicación no se detenga - Verificar que se pueda seguir usando la aplicación normalmente.	
Resultado esperado: al ejecutar la petición de consumo del servicio web de sincronización en segundo plano este no detendrá la aplicación en caso de tardarse en traer una respuesta, así pase más de 5 segundos.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** 7s

Realizado por: Wilmer B. 2018.

Sprint 10

Tabla historia de usuario 18.

HISTORIA DE USUARIO		
Id: HU_18	**Nombre:** Implementación de interfaz de instalación	
Descripción: Como desarrollador necesito crear una interfaz de instalación en donde nos permita sincronizar la app antes de usarla.		
Usuario: Wilmer Barrera	**Sprint:** 10	
Fecha inicio: 07/02/18	**Fecha fin:** 14/02/18	**Esfuerzo:** 20

TAREAS DE INGENIERÍA	
Id	**Nombre**
TI_01	Diseño de interfaz de instalación
TI_02	Implementación de funcionalidad de instalación y primera ejecución de app móvil

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia de usuario 18.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** diseño	**Sprint:** 10
Nombre de historia usuario: HU_18 Implementación de interfaz de instalación		
Nombre tarea: Diseño de interfaz de instalación.		
Fecha inicio: 07/02/18		**Fecha fin:**08/02/18
Descripción:		

Como desarrollador necesito crear una interfaz de instalación en la que se ejecutará los prerrequisitos a ejecutarse antes de usar la aplicación.	
PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_174	Verificar que la interfaz esté acorde con lo requerido por el cliente

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 174.

PRUEBA DE ACEPTACIÓN	
Id: PA_174	**Nombre:** Verificar que la interfaz esté acorde con lo requerido por el cliente.
Tarea de ingeniería: TI_01 Diseño de interfaz de instalación.	
Descripción: La interfaz de instalación debe seguir el bosquejo provisto por el cliente, el mismo que se basará en el color y el formato del texto	
Responsable: Wilmer Barrera	**Fecha:** 08/02/18
Condición de ejecución: - Tener instalado el IDE de desarrollo Android Studio - Tener instalada una máquina virtual en la que se probará el sistema - Tener el bosquejo en el que se basará la interfaz de instalación	
Pasos de ejecución: - Verificar que los compontes de interfaz sean distribuidos de acuerdo al bosquejo - Verificar que el color de la interfaz de instalación sea el mismo definido en el bosquejo - Verificar que el texto de la interfaz siga el mismo formato que le del bosquejo.	
Resultado esperado: La interfaz de instalación sigue el formato de texto, colores y distribución de elementos de interfaz.	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 02 de historia de usuario 18.

TAREA DE INGENIERÍA		
Id: TI_02	**Tipo de tarea:** desarrollo	**Sprint:** 10
Nombre de historia usuario: HU_18 Implementación de interfaz de instalación		
Nombre tarea: Implementación de funcionalidad de instalación y primera ejecución de app móvil.		
Fecha inicio: 09/02/18		**Fecha fin:**14/02/18
Descripción:		

Como desarrollador necesito crear la funcionalidad en la que controle la ejecución de la interfaz de instalación solamente hasta sincronizarla a través del servicio web. Para posteriormente poder usar la aplicación.	
PRUEBAS DE ACEPTACIÓN	
Id	**Nombre**
PA_175	Verificar que la interfaz de instalación se ejecute una vez para sincronizar datos, antes de entrar al menú principal

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 175.

PRUEBA DE ACEPTACIÓN	
Id: PA_175	**Nombre:** Verificar que la interfaz de instalación se ejecute una vez para sincronizar datos, antes de entrar al menú principal.
Tarea de ingeniería: TI_02 Implementación de funcionalidad de instalación y primera ejecución de app móvil.	
Descripción: La interfaz de instalación debe ser presentada antes que todo, mientras la app no sea sincronizada, por lo mismo se tendrá que verificar si la sincronización fue realizada o no	
Responsable: Wilmer Barrera	**Fecha:** 14/02/18
Condición de ejecución: - Tener implementada la interfaz de instalación - Tener instalada la app en el emulador	
Pasos de ejecución: - Verificar que la interfaz se ejecute siempre y cuando la app no haya sido sincronizada por primera vez	
Resultado esperado: La interfaz de instalación se mostrará cada vez que se abra la aplicación y esta no haya sido sincronizada por primera vez	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Tabla historia técnica 05.

HISTORIA TÉCNICA	
Id: HT_05	**Nombre:** Desarrollo de manual de usuario
Descripción: Como desarrollador necesito realizar un documento que contenga las instrucciones de cómo usar la aplicación, así como la reacción de la aplicación ante ciertos inconvenientes que se pueden presentar.	
Usuario: Wilmer Barrera	**Sprint:** 10

Fecha inicio: 15/02/18	**Fecha fin:** 22/02/18	**Esfuerzo:** 30
TAREAS DE INGENIERÍA		
Id	**Nombre**	
TI_01	Implementar manual de usuario	

Realizado por: Wilmer B. 2018.

Tabla tarea de ingeniería 01 de historia técnica 05.

TAREA DE INGENIERÍA		
Id: TI_01	**Tipo de tarea:** documentación	**Sprint:** 10
Nombre de historia técnica: HT_05 Desarrollo de manual de usuario		
Nombre tarea: Implementar manual de usuario.		
Fecha inicio: 15/02/18		**Fecha fin:**22/02/18
Descripción: Como desarrollador necesito crear un manual de usuario en el cual los usuarios puedan aprender a usar la aplicación móvil.		
PRUEBAS DE ACEPTACIÓN		
Id	**Nombre**	
PA_176	Verificar que cada uno de los aspectos de la aplicación móvil sean tocados en el manual de usuario, y especificados de manera clara y precisa.	

Realizado por: Wilmer B. 2018.

Tabla prueba de aceptación 176.

PRUEBA DE ACEPTACIÓN	
Id: PA_176	**Nombre:** Verificar que cada uno de los aspectos de la aplicación móvil sean tocados en el manual de usuario, y especificados de manera clara y precisa.
Tarea de ingeniería: TI_01 Implementar manual de usuario.	
Descripción: el manual de usuario debe tener esquemas en los que se explique el funcionamientos y reacción de la app en ciertas situaciones.	
Responsable: Wilmer Barrera	**Fecha:** 22/02/18
Condición de ejecución: - Tener implementada todas las funcionalidades de la app - Tener creado los servicios web a consumir - Tener creada la base de datos local - Tener creadas las interfaces de la app	
Pasos de ejecución: - Verificar que todas las funcionalidades sean presentadas mediante esquemas	

- Verificar que exista narración en cada uno de los esquemas de la app	
Resultado esperado: El manual de usuario contiene los esquemas y descripciones necesarias dentro del mismo, con el cual el usuario puede aprender a utilizar la app	
Evaluación de la prueba: Exitosa	**Tiempo dedicado:** ---

Realizado por: Wilmer B. 2018.

Anexo C: Pruebas automatizadas por sprint.

Sprint 2

PRUEBAS AUTOMÁTICAS SPRINT 2-HISTORIA DE USUARIO 01

Tabla de requisitos en sprint 2 historia de usuario 01.

<table>
<tr><td rowspan="6">REQUISITOS DE TEST SPRINT 2, HU_01</td><td>Pre-Ejecución</td></tr>
<tr><td>Variables:
private ActivityTestRule<Configuracion> reglaActivity;
private Configuracion mngActivity;
private Context contexto;
private BaseDatos baseDatos;

Función:

@Before
public void setUp() throws Exception {
reglaActivity = new ActivityTestRule<>(Configuracion.class);
reglaActivity.launchActivity(null);
mngActivity = reglaActivity.getActivity();
reglaActivity = new ActivityTestRule<>(Configuracion.class);
contexto= InstrumentationRegistry.getTargetContext();
baseDatos=new BaseDatos(contexto);
}</td></tr>
<tr><td>Métodos Colaboradores</td></tr>
<tr><td>No existen métodos colaboradores</td></tr>
<tr><td>Post Ejecución</td></tr>
<tr><td>Función:

@After
public void tearDown() throws Exception {
baseDatos.close();
}</td></tr>
</table>

Realizado por: Wilmer B. 2018.

Caso de prueba 01 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_01	**Descripción:** Verificar que la base de datos local haya sido creada

Implementación de Test

```
@Test
public void baseDatosCreada() throws Exception {
    assertNotNull(baseDatos);
}
```

Resultado Esperado

La variable **baseDatos** diferente de valor **null**

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 02 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_02	**Descripción:** Verificar eliminación de tabla COOPERATIVA

Implementación de Test

```
@Test
public void eliminarTablaCOOPERATIVA() throws Exception {
    String sql="drop table if exists COOPERATIVA";
    SQLiteDatabase db=baseDatos.getWritableDatabase();
    boolean resp=baseDatos.eliminarTabla(db,sql);
    assertTrue(resp);
}
```

Resultado Esperado

Ejecución de método eliminarTabla(db,sql), retorna valor verdadero (**true**)

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 03 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_03	**Descripción:** Verificar eliminación de tabla DEPENDENCIA

Implementación de Test

```
@Test
public void eliminarTablaDEPENDENCIA() throws Exception {
   String sql="drop table if exists DEPENDENCIA";
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   boolean resp=baseDatos.eliminarTabla(db,sql);
   assertTrue(resp);
}
```

Resultado Esperado

Ejecución de método eliminarTabla(db,sql), retorna valor verdadero (**true**)

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 04 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_04	**Descripción:** Verificar eliminación de tabla DEPENDENCIA POSEE RUTAS

Implementación de Test

```
@Test
public void eliminarTablaDEPEN_POSEE_RUTAS() throws Exception {
   String sql="drop table if exists DEPEN_POSEE_RUTAS";
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   boolean resp=baseDatos.eliminarTabla(db,sql);
   assertTrue(resp);
}
```

Resultado Esperado

Ejecución de método eliminarTabla(db,sql), retorna valor verdadero (**true**)

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 05 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_05	**Descripción:** Verificar eliminación de tabla HORARIO

Implementación de Test

```
@Test
public void eliminarTablaHORARIO() throws Exception {
    String sql="drop table if exists HORARIO";
    SQLiteDatabase db=baseDatos.getWritableDatabase();
    boolean resp=baseDatos.eliminarTabla(db,sql);
    assertTrue(resp);
}
```

Resultado Esperado

Ejecución de método eliminarTabla(db,sql), retorna valor verdadero (**true**)

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 26ms

Realizado por: Wilmer B. 2018.

Caso de prueba 06 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_06	**Descripción:** Verificar eliminación de tabla RUTA

Implementación de Test

```
@Test
public void eliminarTablaRUTA() throws Exception {
    String sql="drop table if exists RUTA";
    SQLiteDatabase db=baseDatos.getWritableDatabase();
    boolean resp=baseDatos.eliminarTabla(db,sql);
    assertTrue(resp);
}
```

Resultado Esperado

Ejecución de método eliminarTabla(db,sql), retorna valor verdadero (**true**)

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 50ms

Realizado por: Wilmer B. 2018.

Caso de prueba 07 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_07	**Descripción:** Verificar eliminación de tabla TERMINAL

Implementación de Test

```
@Test
public void eliminarTablaTERMINAL() throws Exception {
   String sql="drop table if exists TERMINAL";
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   boolean resp=baseDatos.eliminarTabla(db,sql);
   assertTrue(resp);
}
```

Resultado Esperado

Ejecución de método eliminarTabla(db,sql), retorna valor verdadero (**true**)

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 08 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_08	**Descripción:** Verificar eliminación de tabla TERMINAL PERTENCE DEPENDENCIA

Implementación de Test

```
@Test
public void eliminarTablaTERM_PERTENECE_DEPEN() {
   String sql="drop table if exists TERM_PERTENECE_DEPEN";
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   boolean resp=baseDatos.eliminarTabla(db,sql);
   assertTrue(resp);
}
```

Resultado Esperado

Ejecución de método eliminarTabla(db,sql), retorna valor verdadero (**true**)

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 09 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_09	**Descripción:** Verificar eliminación de tabla PROVINCIAS

Implementación de Test

```
@Test
public void eliminarTablaPROVINCIAS() {
   String sql="drop table if exists PROVINCIAS";
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   boolean resp=baseDatos.eliminarTabla(db,sql);
   assertTrue(resp);
}
```

Resultado Esperado

Ejecución de método eliminarTabla(db,sql), retorna valor verdadero (**true**)

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 10 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_10	**Descripción:** Verificar eliminación de tabla CIUDADES

Implementación de Test

```
@Test
public void eliminarTablaCIUDADES() {
   String sql="drop table if exists CIUDADES";
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   boolean resp=baseDatos.eliminarTabla(db,sql);
   assertTrue(resp);
}
```

Resultado Esperado

Ejecución de método eliminarTabla(db,sql), retorna valor verdadero (**true**)

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 11 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_11	**Descripción:** Verificar eliminación de tabla FAVORITO

Implementación de Test

```
@Test
public void eliminarTablaFAVORITO() {
   String sql="drop table if exists FAVORITO";
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   boolean resp=baseDatos.eliminarTabla(db,sql);
   assertTrue(resp);
}
```

Resultado Esperado

Ejecución de método eliminarTabla(db,sql), retorna valor verdadero (**true**)

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 50ms

Realizado por: Wilmer B. 2018.

Caso de prueba 12 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_12	**Descripción:** Verificar eliminación del contenido existente en tabla COOPERATIVA en base de datos local

Implementación de Test

```
@Test
public void eliminarContenidoTablaCOOPERATIVA() {
   String sql="DELETE FROM COOPERATIVA";
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   boolean resp=baseDatos.eliminarContenidoTabla(db,sql);
   assertTrue(resp);
}
```

Resultado Esperado

Ejecución de método eliminarContenidoTabla(db,sql), retorna valor verdadero (**true**)

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 357ms

Realizado por: Wilmer B. 2018.

Caso de prueba 13 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_13	**Descripción:** Verificar eliminación del contenido existente en tabla DEPENDENCIA en base de datos local

Implementación de Test

```
@Test
public void eliminarContenidoTablaDEPENDENCIA() {
   String sql="DELETE FROM DEPENDENCIA";
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   boolean resp=baseDatos.eliminarContenidoTabla(db,sql);
   assertTrue(resp);
}
```

Resultado Esperado

Ejecución de método eliminarContenidoTabla(db,sql), retorna valor verdadero (**true**)

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 586ms

Realizado por: Wilmer B. 2018.

Caso de prueba 14 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_14	**Descripción:** Verificar eliminación del contenido existente en tabla DEPENDENCIA POSEE RUTAS en base de datos local

Implementación de Test

```
@Test
public void eliminarContenidoTablaDEPEN_POSEE_RUTAS() {
   String sql="DELETE FROM DEPEN_POSEE_RUTAS";
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   boolean resp=baseDatos.eliminarContenidoTabla(db,sql);
   assertTrue(resp);
}
```

Resultado Esperado

Ejecución de método eliminarContenidoTabla(db,sql), retorna valor verdadero (**true**)

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 332ms

Realizado por: Wilmer B. 2018.

Caso de prueba 15 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_15	**Descripción:** Verificar eliminación del contenido existente en tabla HORARIO en base de datos local

Implementación de Test

```
@Test
public void eliminarContenidoTablaHORARIO() {
   String sql="DELETE FROM HORARIO";
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   boolean resp=baseDatos.eliminarContenidoTabla(db,sql);
   assertTrue(resp);
}
```

Resultado Esperado

Ejecución de método eliminarContenidoTabla(db,sql), retorna valor verdadero (**true**)

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 381ms

Realizado por: Wilmer B. 2018.

Caso de prueba 16 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_16	**Descripción:** Verificar eliminación del contenido existente en tabla RUTA en base de datos local

Implementación de Test

```
@Test
public void eliminarContenidoTablaRUTA() {
   String sql="DELETE FROM RUTA";
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   boolean resp=baseDatos.eliminarContenidoTabla(db,sql);
   assertTrue(resp);
}
```

Resultado Esperado	
Ejecución de método eliminarContenidoTabla(db,sql), retorna valor verdadero (**true**)	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 335ms

Realizado por: Wilmer B. 2018.

Caso de prueba 17 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_17	**Descripción:** Verificar eliminación del contenido existente en tabla TERMINAL en base de datos local

Implementación de Test

```
@Test
public void eliminarContenidoTablaTERMINAL() {
   String sql="DELETE FROM TERMINAL";
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   boolean resp=baseDatos.eliminarContenidoTabla(db,sql);
   assertTrue(resp);
}
```

Resultado Esperado	
Ejecución de método eliminarContenidoTabla(db,sql), retorna valor verdadero (**true**)	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 458ms

Realizado por: Wilmer B. 2018.

Caso de prueba 18 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_18	**Descripción:** Verificar eliminación del contenido existente en tabla TERMINAL PERTENCE DEPENDENCIA en base de datos local

Implementación de Test

```
@Test
public void eliminarContenidoTablaTERM_PERTENECE_DEPEN() {
   String sql="DELETE FROM TERM_PERTENECE_DEPEN";
   SQLiteDatabase db=baseDatos.getWritableDatabase();
```

boolean resp=**baseDatos**.eliminarContenidoTabla(db,sql); *assertTrue*(resp); }
Resultado Esperado
Ejecución de método eliminarContenidoTabla(db,sql), retorna valor verdadero (**true**)

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 532ms

Realizado por: Wilmer B. 2018.

Caso de prueba 19 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_19	**Descripción:** Verificar eliminación del contenido existente en tabla PROVINCIAS en base de datos local
Implementación de Test	

@Test
public void eliminarContenidoTablaPROVINCIAS() {
String sql=**"DELETE FROM PROVINCIAS"**;
SQLiteDatabase db=**baseDatos**.getWritableDatabase();
boolean resp=**baseDatos**.eliminarContenidoTabla(db,sql);
assertTrue(resp);
}	

Resultado Esperado	
Ejecución de método eliminarContenidoTabla(db,sql), retorna valor verdadero (**true**)	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 382ms

Realizado por: Wilmer B. 2018.

Caso de prueba 20 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_20	**Descripción:** Verificar eliminación del contenido existente en tabla CIUDADES en base de datos local
Implementación de Test	

```
@Test
public void eliminarContenidoTablaCIUDADES() {
   String sql="DELETE FROM CIUDADES";
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   boolean resp=baseDatos.eliminarContenidoTabla(db,sql);
   assertTrue(resp);
}
```

Resultado Esperado	
Ejecución de método eliminarContenidoTabla(db,sql), retorna valor verdadero (**true**)	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 484ms

Realizado por: Wilmer B. 2018.

Caso de prueba 21 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_21	**Descripción:** Verificar eliminación del contenido existente en tabla FAVORITO en base de datos local

Implementación de Test

```
@Test
public void eliminarContenidoTablaFAVORITO() {
   String sql="DELETE FROM FAVORITO";
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   boolean resp=baseDatos.eliminarContenidoTabla(db,sql);
   assertTrue(resp);
}
```

Resultado Esperado	
Ejecución de método eliminarContenidoTabla(db,sql), retorna valor verdadero (**true**)	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 506ms

Realizado por: Wilmer B. 2018.

Caso de prueba 22 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_22	**Descripción:** Verificar la creación de tabla COOPERATIVA en base de datos local

Implementación de Test

```
@Test
public void crearTablaCOOPERATIVA() throws Exception
{
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   baseDatos.eliminarTabla(db,"drop table if exists COOPERATIVA");
   String queryTabCooperativa="CREATE TABLE COOPERATIVA (" +
         "  id_COP INTEGER PRIMARY KEY," +
         "  nombre_COP TEXT," +
         "  pagina_web_COP TEXT);";
   boolean resp=baseDatos.crearTabla(db,queryTabCooperativa);
   assertEquals(true,resp);
}
```

Resultado Esperado

Al ejecutar método crearTabla(db,queryTabCooperativa) debe retornar un valor positivo (**true**).

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 50ms

Realizado por: Wilmer B. 2018.

Caso de prueba 23 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_23	**Descripción:** Verificar la creación de tabla DEPENDENCIA en base de datos local

Implementación de Test

```
@Test
public void crearTablaDEPENDENCIA() throws Exception {
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   baseDatos.eliminarTabla(db,"drop table if exists DEPENDENCIA");
```

```
    String queryTabDependencia="CREATE TABLE DEPENDENCIA (" +
            "  id_DEP INTEGER PRIMARY KEY," +
            "  nombre_DEP TEXT," +
            "  telefono_DEP TEXT," +
            "  correo_DEP TEXT," +
            "  ciudad_DEP TEXT," +
            "  provincia_DEP TEXT," +
            "  direccion_DEP TEXT," +
            "  dias_laborables_DEP INTEGER," +
            "  id_COP INTEGER," +
            "  FOREIGN KEY (id_COP)" +
            "  REFERENCES COOPERATIVA (id_COP));";
    boolean resp=baseDatos.crearTabla(db,queryTabDependencia);
    assertEquals(true,resp);
}
```

Resultado Esperado	
Al ejecutar método crearTabla(db,queryTabDependencia) debe retornar un valor positivo (**true**).	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 24 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_24	**Descripción:** Verificar la creación de tabla DEPENDENCIA POSEE RUTAS en base de datos local

Implementación de Test

```
@Test
public void crearTablaDEPEN_POSEE_RUTAS() throws Exception {
    SQLiteDatabase db=baseDatos.getWritableDatabase();
    baseDatos.eliminarTabla(db,"drop table if exists DEPEN_POSEE_RUTAS");
    String queryTabDPR="CREATE TABLE DEPEN_POSEE_RUTAS (" +
            "  id_DPR INTEGER," +
            "  turno_DPR INTEGER," +
            "  costo_DPR DOUBLE," +
```

```
        " tiempo_DPR TEXT," +
        " id_RTA INTEGER," +
        " id_DEP INTEGER," +
        " PRIMARY KEY (id_DPR,id_RTA,id_DEP)," +
        " FOREIGN KEY (id_RTA)" +
        " REFERENCES RUTA (id_RTA)," +
        " FOREIGN KEY (id_DEP)" +
        " REFERENCES DEPENDENCIA (id_DEP));";
    boolean resp=baseDatos.crearTabla(db,queryTabDPR);
    assertEquals(true,resp);
}
```

Resultado Esperado	
Al ejecutar método crearTabla(db,queryTabDPR) debe retornar un valor positivo (**true**).	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 25 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_25	**Descripción:** Verificar la creación de tabla HORARIO en base de datos local

Implementación de Test

```
@Test
public void crearTablaHORARIO() throws Exception
{
    SQLiteDatabase db=baseDatos.getWritableDatabase();
    baseDatos.eliminarTabla(db,"drop table if exists HORARIO");
    String queryTabHorario="CREATE TABLE HORARIO (" +
        " id_HRR INTEGER PRIMARY KEY," +
        " hora_salida_HRR TEXT," +
        " id_DPR INTEGER," +
        " FOREIGN KEY (id_DPR)" +
        " REFERENCES DEPEN_POSEE_RUTAS (id_DPR));";
    boolean resp=baseDatos.crearTabla(db,queryTabHorario);
```

```
    assertEquals(true,resp);
}
```

Resultado Esperado	
Al ejecutar método crearTabla(db,queryTabHorario) debe retornar un valor positivo (**true**).	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 51ms

Realizado por: Wilmer B. 2018.

Caso de prueba 26 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_26	**Descripción:** Verificar la creación de tabla RUTA en base de datos local

Implementación de Test

```
@Test
public void crearTablaRUTA() throws Exception
{
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   baseDatos.eliminarTabla(db,"drop table if exists RUTA");
   String queryTabRuta="CREATE TABLE RUTA (" +
         "  id_RTA INTEGER PRIMARY KEY," +
         "  ciudad_origen_RTA TEXT," +
         "  ciudad_destino_RTA TEXT);";
   boolean resp=baseDatos.crearTabla(db,queryTabRuta);
   assertEquals(true,resp);
}
```

Resultado Esperado	
Al ejecutar método crearTabla(db,queryTabRuta) debe retornar un valor positivo (**true**).	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 50ms

Realizado por: Wilmer B. 2018.

Caso de prueba 27 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_27	**Descripción:** Verificar la creación de tabla TERMINAL en base de datos local

Implementación de Test

```
@Test
public void crearTablaTERMINAL() throws Exception
{
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   baseDatos.eliminarTabla(db,"drop table if exists TERMINAL");
   String queryTabTerminal="CREATE TABLE TERMINAL (" +
         "  id_TRM INTEGER PRIMARY KEY," +
         "  nombre_TRM TEXT," +
         "  direccion_TRM TEXT);";
   boolean resp=baseDatos.crearTabla(db,queryTabTerminal);
   assertEquals(true,resp);
}
```

Resultado Esperado

Al ejecutar método crearTabla(db,queryTabTerminal) debe retornar un valor positivo (**true**).

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 75ms

Realizado por: Wilmer B. 2018.

Caso de prueba 28 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_28	**Descripción:** Verificar la creación de tabla TERMINAL PERTENCE DEPENDENCIA en base de datos local

Implementación de Test

```
@Test
public void crearTablaTERM_PERTENECE_DEPEN() throws Exception
{
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   baseDatos.eliminarTabla(db,"drop table if exists TERM_PERTENECE_DEPEN");
```

```
    String queryTabTPD="CREATE TABLE TERM_PERTENECE_DEPEN (" +
            "  id_DEP INTEGER PRIMARY KEY," +
            "  id_TRM INTEGER," +
            "  FOREIGN KEY (id_DEP)" +
            "  REFERENCES DEPENDENCIA (id_DEP)," +
            "  FOREIGN KEY (id_TRM)" +
            "  REFERENCES TERMINAL (id_TRM));";
    boolean resp=baseDatos.crearTabla(db,queryTabTPD);
    assertEquals(true,resp);
}
```

Resultado Esperado	
Al ejecutar método crearTabla(db,queryTabTPD) debe retornar un valor positivo (**true**).	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 75ms

Realizado por: Wilmer B. 2018.

Caso de prueba 29 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_29	**Descripción:** Verificar la creación de tabla PROVINCIAS en base de datos local

Implementación de Test

```
@Test
public void crearTablaPROVINCIAS() throws Exception
{
    SQLiteDatabase db=baseDatos.getWritableDatabase();
    baseDatos.eliminarTabla(db,"drop table if exists PROVINCIAS");
    String queryTabProvicia="CREATE TABLE PROVINCIAS (" +
            "  id_PRV INTEGER PRIMARY KEY," +
            "  nombre_PRV TEXT );";
    boolean resp=baseDatos.crearTabla(db,queryTabProvicia);
    assertEquals(true,resp);
}
```

Resultado Esperado	
Al ejecutar método crearTabla(db,queryTabProvincia) debe retornar un valor positivo (**true**).	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 51ms

Realizado por: Wilmer B. 2018.

Caso de prueba 30 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_30	**Descripción:** Verificar la creación de tabla CIUDADES en base de datos local

Implementación de Test

```
@Test
public void crearTablaCIUDADES() throws Exception
{
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   baseDatos.eliminarTabla(db,"drop table if exists CIUDADES");
   String queryTabCiudad="CREATE TABLE CIUDADES (" +
         "  id_CUD INTEGER PRIMARY KEY," +
         "  nombre_CUD TEXT," +
         "  latitud_CUD DOUBLE," +
         "  longitud_CUD DOUBLE," +
         "  id_PRV INTEGER," +
         "  FOREIGN KEY (id_PRV)" +
         "  REFERENCES PROVINCIAS (id_PRV));";
   boolean resp=baseDatos.crearTabla(db,queryTabCiudad);
   assertEquals(true,resp);
}
```

Resultado Esperado	
Al ejecutar método crearTabla(db,queryTabCiudad) debe retornar un valor positivo (**true**).	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 75ms

Realizado por: Wilmer B. 2018.

Caso de prueba 31 de historia de usuario 01.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 02	**Funcionalidad:** HU_01 Codificación de scripts SQL para transacciones en base de datos
Código: TC_31	**Descripción:** Verificar la creación de tabla FAVORITO en base de datos local

Implementación de Test

```
@Test
public void crearTablaFAVORITO() throws Exception
{
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   baseDatos.eliminarTabla(db,"drop table if exists FAVORITO");
   String queryTabFavorito="CREATE TABLE IF NOT EXISTS FAVORITO (" +
         "  id_FAV INTEGER," +
         "  origen_FAV TEXT," +
         "  destino_FAV TEXT," +
         "  transporte_FAV TEXT," +
         "  tiempo_estimado_FAV TEXT," +
         "  turno_FAV TEXT," +
         "  costo_FAV DOUBLE," +
         "  hora_FAV TEXT," +
         "  fecha_FAV TEXT );";
   boolean resp=baseDatos.crearTabla(db,queryTabFavorito);
   assertEquals(true,resp);
}
```

Resultado Esperado

Al ejecutar método crearTabla(db,queryTabFavorito) debe retornar un valor positivo (**true**).

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 51ms

Realizado por: Wilmer B. 2018.

Tabla de resultados en pruebas automáticas del sprint 2 historia de usuario 01.

RESULTADO	
Cantidad de pruebas	31
Tiempo total en ejecución de pruebas Sprint 2, HU_01	16s 916ms

Realizado por: Wilmer B. 2018.

Sprint 3

PRUEBAS AUTOMÁTICAS SPRINT 3-HISTORIA DE USUARIO 03

Caso de prueba 01 de historia de usuario 03.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 03	**Funcionalidad:** HU_03 Implementación de servicios web Geocoding API de Google Maps a través de petición GET
Código: TC_01	**Descripción:** Verificar que haya respuesta del consumo de servicio web de Google Maps

Entrada(s)

```
@Test
public void  consumoSWGeocodingconResultado()
{
   String lat="-1.6732642";
   String lon="-78.6581558";
   String respuesta=new Geolocalizacion().peticionSWgoogleMaps(lat,lon);
   assertNotEquals(respuesta,null);
}
```

Resultado Esperado

Al consumir el servicio web geocoding, debe existir respuesta o sea diferente a **null**

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 708ms

Realizado por: Wilmer B. 2018.

Caso de prueba 02 de historia de usuario 03.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 03	**Funcionalidad:** HU_03 Implementación de servicios web Geocoding API de Google Maps a través de petición GET
Código: TC_02	**Descripción:** Chequear cuando el consumo de servicio web geocoding no fue exitoso

Entrada(s)

```
@Test
public void respuestaVaciaSWGeocoding()
{
   String respuesta=new Geolocalizacion().peticionSWgoogleMaps("","");
```

```
    assertEquals(respuesta,null);
}
```

Resultado Esperado	
Al no existe respuesta de servicio web geocoding la respuesta será un valor **null**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 188ms

Realizado por: Wilmer B. 2018.

Caso de prueba 03 de historia de usuario 03.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 03	**Funcionalidad:** HU_03 Implementación de servicios web Geocoding API de Google Maps a través de petición GET
Código: TC_03	**Descripción:** Chequear cuando consumo de servicio web geocoding fue exitosa pero no obtuvo resultados

Entrada(s)

```
@Test
public void consumoSWGeocodingSinResultados()
{
   String respuesta= null;
   try {
      String lat="7.423021";
      String lon="93.083739";
      Geolocalizacion geo= new Geolocalizacion();
      String respGeo=geo.peticionSWgoogleMaps(lat,lon);
      respuesta = (new JSONObject(respGeo)).getString("status");
   } catch (JSONException e) {
      e.printStackTrace();
   }
   assertEquals(respuesta,"ZERO_RESULTS");
}
```

Resultado Esperado	
Cuando existe consumo exitoso de servicio web geocoding, pero no encuentra resultados de direcciones en base a la latitud y longitud esta devuelve un objeto json con un estado **"ZERO_RESULTS"**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 2s 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 04 de historia de usuario 03.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 03	**Funcionalidad:** HU_03 Implementación de servicios web Geocoding API de Google Maps a través de petición GET
Código: TC_04	**Descripción:** Verificar consumo de servicio web geocoding sea exitoso

Entrada(s)

```
@Test
public void estadoRespuestaConsumoCorrecto()
{
   String respuesta= null;
   try {
      String lat="-4.105316";
      String lon="-80.204493";
      Geolocalizacion geo= new Geolocalizacion();
      String respGeo=geo.peticionSWgoogleMaps(lat,lon);
      respuesta = (new JSONObject(respGeo)).getString("status");
   } catch (JSONException e) {
      e.printStackTrace();
   }
   assertEquals(respuesta,"OK");
}
```

Resultado Esperado

Cuando existe consumo exitoso de servicio web geocoding y además trae lista de direcciones en base a la latitud y longitud, viene con un estado **"OK"**

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 634ms

Realizado por: Wilmer B. 2018.

Caso de prueba 05 de historia de usuario 03.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 03	**Funcionalidad:** HU_03 Implementación de servicios web Geocoding API de Google Maps a través de petición GET
Código: TC_05	**Descripción:** Verificar que la respuesta del consumo de servicio web geocoding obtuvo una lista de resultados

Entrada(s)

```
@Test
public void cosumoSWConListaResultados()
{
   String respuesta= null;
   try {
      String lat="7.423021";
      String lon="93.083739";
      Geolocalizacion geo= new Geolocalizacion();
      String respGeo=geo.peticionSWgoogleMaps(lat,lon);
      respuesta = (new JSONObject(respGeo)).getString("results");
   } catch (JSONException e) {
      e.printStackTrace();
   }
   assertNotEquals(respuesta,null);
}
```

Resultado Esperado	
Al consumir el servicio web de manera exitosa, tiene una lista de direcciones que vienen en un campo **"results"** que es diferente de **null**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 2s 780ms

Realizado por: Wilmer B. 2018.

Caso de prueba 06 de historia de usuario 03.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 03	**Funcionalidad:** HU_03 Implementación de servicios web Geocoding API de Google Maps a través de petición GET
Código: TC_06	**Descripción:** Verificar que la conexión http es aceptada

Entrada(s)

```
@Test
public void conexionHttpAceptada()
{
   try {
      String latLon="-4.105316,-80.204493";
      URL url = new URL(ConsSW.urlSWgoogMaps+latLon+"&language=es");
      HttpURLConnection connection = (HttpURLConnection) url.openConnection();
      int codResp=connection.getResponseCode();
      boolean respuesta=new Geolocalizacion().establecioConexionHttp(codResp);
```

```
        assertEquals(true,respuesta);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Resultado Esperado

Cuando se abre una petición http para consumir servicio web geocoding, si la conexión con dicho servicio web es aceptada será un valor **true**

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 456ms

Realizado por: Wilmer B. 2018.

Caso de prueba 07 de historia de usuario 03.

CASO DE PRUEBA AUTOMATIZADA

Sprint No: 03	**Funcionalidad:** HU_03 Implementación de servicios web Geocoding API de Google Maps a través de petición GET
Código: TC_07	**Descripción:** Chequear que la conexión htttp fue rechazada

Entrada(s)

```
@Test
public void conexionHttpRechazada()
{
    try {
        String latLon="-4.105316,-80.204493";
        URL url = new URL(ConsSW.urlSWgoogMaps+latLon+"&language=es");
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        int codResp=404;
        boolean respuesta=new Geolocalizacion().establecioConexionHttp(codResp);
        assertEquals(false,respuesta);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Resultado Esperado

Cuando se abre una petición http, para consumir servicio web geocoding, si la conexión con dicho servicio web es rechazada, será un valor **false**

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Tabla de resultados en pruebas automáticas del sprint 3 historia de usuario 03.

RESULTADO	
Cantidad de pruebas	7
Tiempo total en ejecución de pruebas Sprint 3, HU_03	7s 791ms

Realizado por: Wilmer B. 2018.

Sprint 4

PRUEBAS AUTOMÁTICAS SPRINT 4-HISTORIA DE USUARIO 04

Tabla de requisitos en sprint 4 historia de usuario 04.

REQUISITOS DE TEST SPRINT 4, HU_04	**Pre-Ejecución**

	Variables:
	private Context **contexto**;
	private BaseDatos **baseDatos**;
	private ArrayList<Provincia> **lstProvincias**;
	private ArrayList<Ciudad> **lstCiudades**;
	
	Función:
	
	@Before
	public void setUp() **throws** Exception {
	contexto=InstrumentationRegistry.*getTargetContext*();
	baseDatos=**new** BaseDatos(**contexto**);
	}

	Métodos Colaboradores
	public void llenarProvincias() { **lstProvincias**= **new** ArrayList<>(); **lstProvincias**.add(**new** Provincia(1,**"azuay"**)); **lstProvincias**.add(**new** Provincia(2,**"bolívar"**)); **lstProvincias**.add(**new** Provincia(3,**"cañar"**)); **lstProvincias**.add(**new** Provincia(4,**"carchi"**)); **lstProvincias**.add(**new** Provincia(5,**"chimborazo"**)); **lstProvincias**.add(**new** Provincia(6,**"cotopaxi"**)); **lstProvincias**.add(**new** Provincia(7,**"el oro"**));

```
    lstProvincias.add(new Provincia(8,"esmeraldas"));
    lstProvincias.add(new Provincia(9,"galápagos"));
    lstProvincias.add(new Provincia(10,"guayas"));
    lstProvincias.add(new Provincia(11,"santo domingo de los tsáchilas"));
    lstProvincias.add(new Provincia(12,"santa elena"));
    lstProvincias.add(new Provincia(13,"imbabura"));
    lstProvincias.add(new Provincia(14,"loja"));
    lstProvincias.add(new Provincia(15,"los ríos"));
    lstProvincias.add(new Provincia(16,"manabí"));
    lstProvincias.add(new Provincia(17,"morona santiago"));
    lstProvincias.add(new Provincia(18,"napo"));
    lstProvincias.add(new Provincia(19,"orellana"));
    lstProvincias.add(new Provincia(20,"pastaza"));
    lstProvincias.add(new Provincia(21,"pichincha"));
    lstProvincias.add(new Provincia(22,"sucumbíos"));
    lstProvincias.add(new Provincia(23,"tungurahua"));
    lstProvincias.add(new Provincia(24,"zamora chinchipe"));
}

public void llenarCiudades()
{
    lstCiudades= new ArrayList<>();
    lstCiudades.add(new Ciudad("riobamba",-1.6732642,-78.6581558));
    lstCiudades.add(new Ciudad("quito",-0.1568747,-78.540189));
    lstCiudades.add(new Ciudad("guayaquil",-2.1774458,-79.9591627));
    lstCiudades.add(new Ciudad("cuenca",-2.8987054,-78.996889));
    lstCiudades.add(new Ciudad("machala",-3.2544197,-79.9644901));
    lstCiudades.add(new Ciudad("puyo",-1.4892927,-78.009359));
}
```

Post Ejecución

Función:

```
@After
public void tearDown() throws Exception {
    baseDatos.close();
}
```

Realizado por: Wilmer B. 2018.

Caso de prueba 01 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_01	**Descripción:** Chequear que la lista de direcciones filtradas esté vacía

Entrada(s)

```
@Test
public void listaDireccionesVacia()
{
   JSONArray arrayJson= null;
   ArrayList<String> respuesta=null;

   try {
      arrayJson = new JSONArray("[]");
      Geolocalizacion geo=new Geolocalizacion();
      respuesta=geo.filtrarDatosSWgoogleMaps(arrayJson);
   } catch (JSONException e) {
      e.printStackTrace();
   }
   assertEquals(respuesta,null);
}
```

Resultado Esperado

Cuando la lista de direcciones tiene un formato json incorrecto, entonces en el filtrado de las mismas devolverá un valor **null**

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 02 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_02	**Descripción:** Verificar que existe lista de direcciones filtradas

Entrada(s)

```
@Test
public void listaDireccionesConDatos()
{
   JSONArray arrayJson= null;
   ArrayList<String> respuesta=null;

   try {
      Geolocalizacion geo=new Geolocalizacion();
      String lat="-4.105316";
      String lon="-80.204493";
      String respGeo=geo.peticionSWgoogleMaps(lat,lon);
      arrayJson = (new JSONObject(respGeo)).getJSONArray("results");
      respuesta=geo.filtrarDatosSWgoogleMaps(arrayJson);
   } catch (JSONException e) {
      e.printStackTrace();
   }
   assertNotEquals(respuesta,null);
}
```

Resultado Esperado	
Cuando se manda a filtrar la respuesta del servicio web geocoding, este filtro devolverá una lista de datos filtrados que será diferente de **null**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 35ms

Realizado por: Wilmer B. 2018.

Caso de prueba 03 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_03	**Descripción:** Formatear lista de direcciones filtradas

Entrada(s)

```
@Test
public void formatearListaDirecciones()
{
   String[] direcciones=new String[4];
   direcciones[0]="Barón de Carondelet, Riobamba, Ecuador";
   direcciones[1]="Riobamba, Ecuador";
```

```
    direcciones[2]="Provincia de Chimborazo, Ecuador";
    direcciones[3]="Ecuador";

    String[] direccionesEspe=new String[4];
    direccionesEspe[0]="Barón de Carondelet";
    direccionesEspe[1]="Riobamba";
    direccionesEspe[2]="Ecuador";
    direccionesEspe[3]="Provincia de Chimborazo";

    Geolocalizacion geo=new Geolocalizacion();
    ArrayList<String> respuesta=geo.formatearDirecciones(direcciones);
    String[] lstResp=new String[respuesta.size()];
    for (int i=0;i<respuesta.size();i++)
       lstResp[i]=respuesta.get(i);

    assertArrayEquals(lstResp,direccionesEspe);
}
```

Resultado Esperado	
Cuando se manda a formatear una lista de direcciones, entonces está se guardará en una lista de strings evitando la redundancia de las direcciones, el resultado esperado en este caso será: String[] direccionesEspe=new String[4]; direccionesEspe[0]="Barón de Carondelet"; direccionesEspe[1]="Riobamba"; direccionesEspe[2]="Ecuador"; direccionesEspe[3]="Provincia de Chimborazo";	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 04 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_04	**Descripción:** Cargar provincias desde base de datos local en una lista
Entrada(s)	

```
@Test
public void cargarProvincias() throws Exception {
   SincronizarBasesDatos sbd=new SincronizarBasesDatos(contexto,null);
   sbd.realizarPeticion();
   ArrayList<Provincia> resp=baseDatos.cargarProvincias();
   assertEquals(24,resp.size());
}
```

Resultado Esperado	
Cuando se manda a cargar la lista de provincias del Ecuador desde la base de datos local deberá extraer una lista con 24 elementos, correspondientes a todas las provincias del Ecuador	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 05 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_05	**Descripción:** Chequear que la lista de provincias de base de datos local este vacía

Entrada(s)

```
@Test
public void noExistenProvinciasRegistradas() throws Exception {
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   baseDatos.eliminarContenidoTabla(db,"DELETE FROM PROVINCIAS");
   ArrayList<Provincia> resp=baseDatos.cargarProvincias();
   assertEquals(0,resp.size());
}
```

Resultado Esperado	
Cuando se intenta extraer la lista de provincias desde la base de datos local, y no existen aún provincias registradas entonces devolverá una lista con 0 elementos	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 532ms

Realizado por: Wilmer B. 2018.

Caso de prueba 06 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_06	**Descripción:** Encontrar coincidencia de provincia entre la lista de direcciones y la lista de provincias extraídas de la base de datos local

Entrada(s)

```
@Test
public void CoincidenciasListaDireccionesProvincias()
{
   llenarProvincias();
   ArrayList<String> direcciones=new ArrayList<>();
   direcciones.add("barón de carondelet");
   direcciones.add("riobamba");
   direcciones.add("ecuador");
   direcciones.add("provincia de chimborazo");

   Geolocalizacion geo=new Geolocalizacion();
   int idPrv=geo.obtenerProvincia(direcciones,lstProvincias);
   assertEquals(idPrv,5);
}
```

Resultado Esperado

Cuando se tiene una lista de direcciones filtradas y formateadas, entonces se cotejará la lista de direcciones con la lista de provincias, del cual al encontrará una coincidencia devolverá un id de provincia, en este caso 5

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 07 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_07	**Descripción:** Chequear cuando no se encuentra coincidencia de provincia entre la lista de direcciones y la lista de provincias extraídas de la base de datos local

Entrada(s)

```
@Test
public void SinCoincidenciasListaDireccionesProvincias()
{
   llenarProvincias();
   ArrayList<String> direcciones=new ArrayList<>();
   direcciones.add("barón de carondelet");
   direcciones.add("riobamba");
   direcciones.add("ecuador");
   direcciones.add("riobamba");

   Geolocalizacion geo=new Geolocalizacion();
   int idPrv=geo.obtenerProvincia(direcciones,lstProvincias);
   assertEquals(idPrv,-1);
}
```

Resultado Esperado	
Cuando se busca coincidencias de provincias entre la lista de direcciones y la lista de provincias extraída de la base de datos, entonces no se encuentra ningún id de provincia por lo que devolverá un valor de -1	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 08 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_08	**Descripción:** Cargar lista de ciudades desde base de datos local en una lista, en base al id de una provincia

Entrada(s)

```
@Test
public void cargarCiudadesPorIdProvincia() throws Exception {

   ArrayList<Ciudad> resp=baseDatos.cargarCiudades(2);
   assertEquals(true,resp.size()>0);
}
```

Resultado Esperado

Cuando se extrae una lista de ciudades que corresponden a una provincia, esta consulta deberá devolver una lista de ciudades con un número de elementos mayor a 0	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 09 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_09	**Descripción:** Chequear cuando la lista de ciudades extraídas de la base en base al id de una provincia sea vacía

Entrada(s)

```
@Test//filtarr geocding
public void noExistenCiudadPorIdProvincia() throws Exception {
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   baseDatos.eliminarContenidoTabla(db,"DELETE FROM CIUDADES");
   ArrayList<Ciudad> resp=baseDatos.cargarCiudades(2);
   assertEquals(0,resp.size());
}
```

Resultado Esperado

Cuando se extrae una lista de ciudades de una provincia específica y no existen aún ciudades registradas, entonces la consulta devolverá una lista con 0 elementos

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 330ms

Realizado por: Wilmer B. 2018.

Caso de prueba 10 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_10	**Descripción:** Encontrar coincidencia de ciudad entre la lista de direcciones y la lista de ciudades extraídas de la base de datos

Entrada(s)

```
@Test
public void CoincidenciasListaDireccionesCiudades()
{
   llenarCiudades();
```

```
    ArrayList<String> direcciones=new ArrayList<>();
    direcciones.add("barón de carondelet");
    direcciones.add("riobamba");
    direcciones.add("ecuador");
    direcciones.add("provincia de chimborazo");

    Geolocalizacion geo=new Geolocalizacion();
    String ciudad=geo.obtenerCiudad(direcciones,lstCiudades);
    assertEquals(ciudad,"riobamba");
}
```

Resultado Esperado	
Al tener una lista de direcciones de respuesta del servicio web, se manda a cotejar coincidencia de ciudades entre la lista de direcciones y la lista de ciudades extraídas de la base de datos, al encontrar una coincidencia, entonces devolverá dicha ciudad en la que coincidieron, en este caso **"riobamba"**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 11 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_11	**Descripción:** Chequear cuando no se encontraron coincidencias de ciudad entre la lista de direcciones y la lista de ciudades extraídas de la base de datos

Entrada(s)

```
@Test
public void SinCoincidenciasListaDireccionesCiudades()
{
    llenarCiudades();
    ArrayList<String> direcciones=new ArrayList<>();
    direcciones.add("barón de carondelet");
    direcciones.add("baños de cuenca");
    direcciones.add("ecuador");
    direcciones.add("provincia de bolivar");
```

```
    Geolocalizacion geo=new Geolocalizacion();
    String ciudad=geo.obtenerCiudad(direcciones,lstCiudades);
    assertEquals(ciudad,null);
}
```

Resultado Esperado

Al mandar a cotejar coincidencias de ciudad entre la lista de direcciones y la lista de ciudades extraídas desde la base de datos y no existe ninguna coincidencia entonces devolverá un valor **null**

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 12 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA

Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_12	**Descripción:** Obtener distancia entre dos coordenadas gps a través de geometría esférica

Entrada(s)

```
@Test
public void distanciaEntreDosPuntosMetodoEsfera()
{
    Geolocalizacion geo=new Geolocalizacion();
    //Machachi
    double lat1=-0.5133183;
    double lon1=-78.5787494;
    //Quito
    double lat2=-0.1568747;
    double lon2=-78.540189;
    double respuesta=geo.obtenerDistanciaCoodenadas(lat1,lon1,lat2,lon2);
    assertEquals(respuesta,39.87719655930266,5);
}
```

Resultado Esperado

Para buscar la ciudad más cercana en base a coordenadas gps se saca la distancia entre dos puntos esféricamente, la misma que se debe poner un margen de error, en este caso 5 y como respuesta de distancia: 39.87719655930266

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 13 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_13	**Descripción:** Transformar cantidad matemática normal a radianes

Entrada(s)

```
@Test
public void convertirCantidadToRadianes()
{
   Geolocalizacion geo=new Geolocalizacion();
   double respuesta=geo.convertirDistanciaEnRdianes(126.3);
   assertEquals(respuesta,2.2043508452688383,5);
}
```

Resultado Esperado

Para obtener la distancia entre dos puntos esféricamente en base a coordenadas gps, se necesita pasar dicho resultado a radianes, en este caso la respuesta es: 2.2043508452688383

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 26ms

Realizado por: Wilmer B. 2018.

Caso de prueba 14 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_14	**Descripción:** Obtener el índice del número menor de una lista de distancias entre ciudades

Entrada(s)

```
@Test
public void indiceDeMenorEntreDistanciasEntreCiudades()
{
   double[] lstDistancias= new double[5];
   lstDistancias[0]=345.2;
   lstDistancias[1]=35.4;
   lstDistancias[2]=5.256;
   lstDistancias[3]=1565.1;
   lstDistancias[4]=125.0;
```

```
    Geolocalizacion geo=new Geolocalizacion();
    int indice=geo.menorDistancia(lstDistancias);
    assertEquals(2,indice);
}
```

Resultado Esperado	
Cuando se tiene la lista de distancias entre una ciudad específica y todas las demás extraídas de a base de datos, entonces de esta lista se obtendrá el índice de la menor distancia entre las dos ciudades, en este caso: 2	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 26ms

Realizado por: Wilmer B. 2018.

Caso de prueba 15 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_15	**Descripción:** Cargar todas las ciudades desde base de datos local en una lista

Entrada(s)

```
@Test
public void cargarTodasLasCiudades() throws Exception {
    llenarCiudades();
    ArrayList<Ciudad> resp=baseDatos.cargarAllCiudades();
    assertEquals(true,resp.size()>0);
}
```

Resultado Esperado	
Cuando se extrae la lista de todas las ciudades de la base de datos, esta devuelve una lista de ciudades con un número de elementos mayor a 0	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 16 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_16	**Descripción:** Chequear cuando la lista de todas las ciudades extraída de la base de datos local esté vacía

Entrada(s)	

@Test
public void noExistenCiudades() **throws** Exception {
SQLiteDatabase db=**baseDatos**.getWritableDatabase();
baseDatos.eliminarContenidoTabla(db,**"DELETE FROM CIUDADES"**);
ArrayList<Ciudad> resp=**baseDatos**.cargarAllCiudades();
assertEquals(0,resp.size());
}	

Resultado Esperado	
Cuando se intenta extraer lista de ciudades desde la base de datos y estas no han sido registradas entonces como resultado devolverá una lista con un número de elementos igual a 0	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 283ms

Realizado por: Wilmer B. 2018.

Caso de prueba 17 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_17	**Descripción:** Verificar ciudad cercana en base a otra
Entrada(s)	

@Test
public void verificarCiudadCercana()
{
llenarCiudades();
//machachi
Geolocalizacion geo=**new** Geolocalizacion();
String lat=**"-0.5133183"**;
String lon=**"-78.5787494"**;
String ciudad=geo.buscarCiudadCercana(lat,lon,**lstCiudades**);
assertEquals(ciudad,**"quito"**);
}	

Resultado Esperado	
Cuando se busca una ciudad cercana en base a otra, a través de coordendas gps, esta devolverá como resultado dicha ciudad más cercana, en este caso **"quito"**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 18 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_18	**Descripción:** Chequear cuando no existe ciudad cercana en base a otra

Entrada(s)

```
@Test
public void NoExisteCiudadCercana()
{
   Geolocalizacion geo=new Geolocalizacion();
   String lat="-0.5133183";
   String lon="-78.5787494";
   boolean respuesta=geo.buscarCiudadMasCercana(lat,lon,lstCiudades);
   assertEquals(respuesta,false);
}
```

Resultado Esperado

Cuando se ha buscado una ciudad cercana en base a coordenadas gps, y no se encuentra resultado, entonces está devolverá un valor **false** como respuesta

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 19 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_19	**Descripción:** Chuequear cuando si existe ciudad cercana en base a otra

Entrada(s)

```
@Test
public void verificarCiudadCercana()
{
   Geolocalizacion geo=new Geolocalizacion();
   String lat="-0.5133183";
   String lon="-78.5787494";
   boolean respuesta=geo.buscarCiudadMasCercana(lat,lon,lstCiudades);
   assertEquals(respuesta,true);
```

}
Resultado Esperado
Cuando se ha buscado una ciudad cercana en base a coordenadas gps, y se encuentra resultado, entonces está devolverá un valor **true** como respuesta
Evaluación de la prueba: Exitosa | **Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 20 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_20	**Descripción:** Verificar la existencia de una ciudad específica dentro de la base de datos en base a una ciudad ingresada
Entrada(s)	

@Test
public void existeCiudadOrigen() **throws** Exception {
 	
boolean resp=**baseDatos**.existeCiudadOrigen(**"riobamba"**);
assertTrue(resp);
}	

Resultado Esperado	
Cuando se necesita verificar si una ciudad está registrada en la base de datos local, y la misma existe, entonces esta consulta devolverá una respuesta positiva **true**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 26ms

Realizado por: Wilmer B. 2018.

Caso de prueba 21 de historia de usuario 04.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_04 Implementación de funcionalidad para filtrado de datos geo codificados devueltos por Geocoding API de Google Maps
Código: TC_21	**Descripción:** Verificar si no existen ciudad origen, en base a una ciudad ingresada
Entrada(s)	

@Test
public void noExistenCiudadOrigen() **throws** Exception {
boolean resp=**baseDatos**.existeCiudadOrigen(**"salcedo"**);
assertFalse(resp);
}

Resultado Esperado
Cuando se necesita verificar si una ciudad está registrada en la base de datos local, y la misma no existe, entonces esta consulta devolverá una respuesta negativa **false**
Evaluación de la prueba: Exitosa | **Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Tabla de resultados en pruebas automáticas del sprint 4 historia de usuario 04.

RESULTADO	
Cantidad de pruebas	21
Tiempo total en ejecución de pruebas Sprint 4, HU_04	5s 433ms

Realizado por: Wilmer B. 2018.

PRUEBAS AUTOMÁTICAS SPRINT 4-HISTORIA DE USUARIO 06

Tabla de requisitos en sprint 4 historia de usuario 06.

	Pre-Ejecución
REQUISITOS DE TEST SPRINT 4, HU_06	**Variables:** **private** Context **contexto**; **private** BaseDatos **baseDatos**; **public** ActivityTestRule<BusquedaCoop> **reglaActivity**; **private** BusquedaCoop **mngActivity**; **private** ArrayList<Cooperativa> **lstCooperativas**; **private** ArrayList<Ruta> **lstRutas**; **Función:** @Before **public void** setUp() **throws** Exception { **reglaActivity** = **new** ActivityTestRule<>(BusquedaCoop.**class**); **reglaActivity**.launchActivity(**null**); **mngActivity** = **reglaActivity**.getActivity(); **reglaActivity** = **new** ActivityTestRule<>(BusquedaCoop.**class**);

```
    contexto= InstrumentationRegistry.getTargetContext();
    baseDatos=new BaseDatos(contexto);
}
```

Métodos Colaboradores

```
public void llenarCooperativas()
{
    lstCooperativas= new ArrayList<>();
    lstCooperativas.add(new Cooperativa(1,"riobamba"));
    lstCooperativas.add(new Cooperativa(2,"baños"));
    lstCooperativas.add(new Cooperativa(3,"condorazo"));
    lstCooperativas.add(new Cooperativa(4,"patria"));
}

public void llenarRutas()
{
    lstRutas= new ArrayList<>();
    lstRutas.add(new Ruta(1,"quito","guayaquil",1));
    lstRutas.add(new Ruta(2,"riobamba","cuenca",11));
    lstRutas.add(new Ruta(3,"macas","ambato",13));
    lstRutas.add(new Ruta(4,"guayaquil","cuenca",7));
}
```

Post Ejecución

Función:

```
@After
public void tearDown() throws Exception {
    baseDatos.close();
}
```

Realizado por: Wilmer B. 2018.

Caso de prueba 01 de historia de usuario 06.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_06 Implementación de funcionalidad para consulta de horarios por transporte
Código: TC_01	**Descripción:** Cargar lista de cooperativas de transporte desde la base de datos local

Entrada(s)

```
@Test
public void cargarListaCooperativas() {
    ArrayList<Cooperativa> resp=baseDatos.cargarCooperativas();
    assertEquals(true,resp.size() > 0);
}
```

Resultado Esperado

Al extraer la lista de cooperativas de transporte desde la base de datos local, la misma debe tener un número de elementos mayor a 0

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 508ms

Realizado por: Wilmer B. 2018.

Caso de prueba 02 de historia de usuario 06.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_06 Implementación de funcionalidad para consulta de horarios por transporte
Código: TC_02	**Descripción:** Chequear cuando la lista de transportes esté vacía, cuando no se encuentre ninguna cooperativa de transporte

Entrada(s)

```
@Test
public void noExistenDatosDeCooperativas() {
    SQLiteDatabase db=baseDatos.getWritableDatabase();
    baseDatos.eliminarContenidoTabla(db,"DELETE FROM COOPERATIVA");
    ArrayList<Cooperativa> resp=baseDatos.cargarCooperativas();
    assertEquals(0,resp.size());
}
```

Resultado Esperado

Cuando se intenta extraer lista de cooperativas de transporte desde la base de datos y no existen registros, el resultado que devolverá será una lista de cooperativas con un número de elementos igual a 0

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 462ms

Realizado por: Wilmer B. 2018.

Caso de prueba 03 de historia de usuario 06.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_06 Implementación de funcionalidad para consulta de horarios por transporte
Código: TC_03	**Descripción:** Convertir lista de objetos de cooperativas de transporte a lista de strings

Entrada(s)

```
@Test
public void convertirListaObjsCoopToString() {
   llenarCooperativas();
   String[] lstNombres=new String[4];
   lstNombres[0]="Riobamba";
   lstNombres[1]="Baños";
   lstNombres[2]="Condorazo";
   lstNombres[3]="Patria";

   String[] respuest;
   respuest= mngActivity.convertirLstObjsCoopString(lstCooperativas);
   assertArrayEquals(respuest,lstNombres);
}
```

Resultado Esperado

Al mandar a convertir un objeto de cooperativa para posteriormente presentarlo, este objeto se convertirá en una lista de strins con los nombres de cooperativas, en este caso:

```
String[] lstNombres=new String[4];
   lstNombres[0]="Riobamba";
   lstNombres[1]="Baños";
   lstNombres[2]="Condorazo";
   lstNombres[3]="Patria";
```

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 206ms

Realizado por: Wilmer B. 2018.

Caso de prueba 04 de historia de usuario 06.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_06 Implementación de funcionalidad para consulta de horarios por transporte
Código: TC_04	**Descripción:** Cargar lista de rutas en base al id de una cooperativa de transporte desde la base de datos local

Entrada(s)

```
@Test
public void cargarListaRutas() {
    ArrayList<Dependencia> resp=baseDatos.obtenerRutas(1);
    assertEquals(true,resp.size() > 0);
}
```

Resultado Esperado

Cuando se necesita obtener la lista de rutas que pertenecen a una agencia o dependencia, entonces se obtendrá como resultado una lista de rutas mayor a 0

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 157ms

Realizado por: Wilmer B. 2018.

Caso de prueba 05 de historia de usuario 06.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_06 Implementación de funcionalidad para consulta de horarios por transporte
Código: TC_05	**Descripción:** Chequear cuando la lista de rutas en base al id de una cooperativa de transportes esté vacía

Entrada(s)

```
@Test
public void noExistenDatosDeRutas() {
    ArrayList<Dependencia> resp=baseDatos.obtenerRutas(5);
    assertEquals(0,resp.size());
}
```

Resultado Esperado

Al intentar extraer lista de rutas que pertenecen a una dependencia o agencia, y las mismas no están registradas, entonces el resultado devolverá una lista de rutas con un número de 0 elementos

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 434ms

Realizado por: Wilmer B. 2018.

Caso de prueba 06 de historia de usuario 06.

<table>
<tr><th colspan="2">CASO DE PRUEBA AUTOMATIZADA</th></tr>
<tr><td>Sprint No: 04</td><td>Funcionalidad: HU_06 Implementación de funcionalidad para consulta de horarios por transporte</td></tr>
<tr><td>Código: TC_06</td><td>Descripción: Extraer lista de rutas de lista de objetos dependencia a lista de strings ordenadas alfabéticamente</td></tr>
<tr><th colspan="2">Entrada(s)</th></tr>
<tr><td colspan="2">

```
@Test
public void convertirListObjDependenciaRutasStringOrdenAlfabetico()
{
   llenarRutas();
   ArrayList<Dependencia> lstDependencias= new ArrayList<>();
   Dependencia dep= new Dependencia();
   dep.setLstRutas(lstRutas);
   lstDependencias.add(dep);

   String[] lstNombres=new String[4];
   lstNombres[0]="Guayaquil/Cuenca";
   lstNombres[1]="Macas/Ambato";
   lstNombres[2]="Quito/Guayaquil";
   lstNombres[3]="Riobamba/Cuenca";

   String[] respuest;
   respuest= mngActivity.convertirLstObjDependenciasString(lstDependencias);
   assertArrayEquals(lstNombres,respuest);
}
```

</td></tr>
<tr><th colspan="2">Resultado Esperado</th></tr>
<tr><td colspan="2">Para presentar la lista de rutas de una dependencia, la misma se convierte a una lista de cadena de string con la ciudad origen y fin divididas por un símbolo “/”, en este caso la respuesta es:

```
String[] lstNombres=new String[4];
   lstNombres[0]="Guayaquil/Cuenca";
   lstNombres[1]="Macas/Ambato";
   lstNombres[2]="Quito/Guayaquil";
   lstNombres[3]="Riobamba/Cuenca";
```

</td></tr>
<tr><td>Evaluación de la prueba: Exitosa</td><td>Tiempo de ejecución: 1s 333ms</td></tr>
</table>

Realizado por: Wilmer B. 2018.

Caso de prueba 07 de historia de usuario 06.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_06 Implementación de funcionalidad para consulta de horarios por transporte
Código: TC_07	**Descripción:** Obtener id de dependencia posee ruta en base a ciudad origen y destino de una cooperativa de transporte

Entrada(s)

```
@Test
public void obtenerIdDepPoseeRutas() {
   llenarRutas();
   ArrayList<Dependencia> lstDependencias= new ArrayList<>();
   Dependencia dep= new Dependencia();
   dep.setLstRutas(lstRutas);
   lstDependencias.add(dep);

   int respuesta=mngActivity.obtenerId_Dpr_Dep(1,"macas","ambato",lstDependencias);
   assertEquals(13,respuesta);
}
```

Resultado Esperado

Para extraer el id de dependencia, en base a una ruta con ciudad origen y destino, dentro de la lista de dependencias, entonces se obtendrá como resultado el id de la dependencia, en este caso: 13

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 7ms

Realizado por: Wilmer B. 2018.

Caso de prueba 08 de historia de usuario 06.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_06 Implementación de funcionalidad para consulta de horarios por transporte
Código: TC_08	**Descripción:** No se ha obtenido id de dependencia posee ruta en base a ciudad origen y destino de una cooperativa de transporte

Entrada(s)

```
@Test
public void noExisteIdDepPoseeRutas() {
   llenarRutas();
   ArrayList<Dependencia> lstDependencias= new ArrayList<>();
```

```
    Dependencia dep= new Dependencia();
    dep.setLstRutas(lstRutas);
    lstDependencias.add(dep);

    int respuesta=mngActivity.obtenerId_Dpr_Dep(1,"baños","loja",lstDependencias);
    assertEquals(-1,respuesta);
}
```

Resultado Esperado	
Al intentar obtener el id de la dependencia el base a una ruta específica y dicha ruta no se encuentra en la lista de dependencias entonces como resultado se obtendrá un valor de -1	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 81ms

Realizado por: Wilmer B. 2018.

Caso de prueba 09 de historia de usuario 06.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_06 Implementación de funcionalidad para consulta de horarios por transporte
Código: TC_09	**Descripción:** Obtener las dependencias o agencia junto con las rutas, en base al id de la misma

Entrada(s)

```
@Test
public void cargarDependenciaPoseeRuta() {
    Depen_posee_rutas resp=baseDatos.obetenerDepenPoseeRuta(1);
    assertNotNull(resp);
}
```

Resultado Esperado	
Cuando se necesita cargar una dependencia, junto con la lista de rutas que posee, el objeto resultado que es devuelto es diferente de **null**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 108ms

Realizado por: Wilmer B. 2018.

Caso de prueba 10 de historia de usuario 06.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_06 Implementación de funcionalidad para consulta de horarios por transporte
Código: TC_10	**Descripción:** Verificar que no existe información de dependencia o agencia que poseen rutas, en base a su id
Entrada(s)	

```
@Test
public void noExisteDependenciaPoseeRuta() {
    Depen_posee_rutas resp=baseDatos.obetenerDepenPoseeRuta(105);
    assertNull(resp);
}
```

Resultado Esperado	
Al intentar extraer una dependencia específica, junto con sus rutas y la misma no está registrada entonces, el objeto que se devolverá como resultado será un valor **null**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 856ms

Realizado por: Wilmer B. 2018.

Caso de prueba 11 de historia de usuario 06.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_06 Implementación de funcionalidad para consulta de horarios por transporte
Código: TC_11	**Descripción:** Obtener lista de horarios de una ruta en base al id de Dependencia posee ruta
Entrada(s)	

```
@Test
public void cargarHorariosPorIdDPR() {
    String[][] resp=baseDatos.cargarHorariosRutas(8);
    assertNotNull(resp);
}
```

Resultado Esperado	
Para cargar la lista de horarios de una ruta que pertenece a una dependencia desde la base de datos, si existen dichos horarios, se devolverá como resultado una matriz con los horarios extraídos, la misma que será diferente a un valor **null**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 409ms

Realizado por: Wilmer B. 2018.

Caso de prueba 12 de historia de usuario 06.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 04	**Funcionalidad:** HU_06 Implementación de funcionalidad para consulta de horarios por transporte
Código: TC_12	**Descripción:** Verificar que no existan datos registrados de los horarios de una ruta en base al id de dependencia posee ruta
Entrada(s)	

@Test
public void noExisteHorariosPorIdDPR() {
String[][] resp=**baseDatos**.cargarHorariosRutas(105);
assertNull(resp);
}	

Resultado Esperado	
Al intentar extraer la lista de horarios de una ruta que pertenece a una dependencia y dicha ruta aún no posee horarios, entonces se obtendrá como respuesta una matriz con valor **null**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 132ms

Realizado por: Wilmer B. 2018.

Tabla de resultados en pruebas automáticas del sprint 4 historia de usuario 06.

RESULTADO	
Cantidad de pruebas	12
Tiempo total en ejecución de pruebas Sprint 4, HU_06	14s 693ms

Realizado por: Wilmer B. 2018.

Sprint 5

PRUEBAS AUTOMÁTICAS SPRINT 5-HISTORIA DE USUARIO 08

Tabla de requisitos en sprint 5 historia de usuario 08.

REQUISITOS DE TEST SPRINT 5 HU_08	**Pre-Ejecución**
	Variables: **private** Context **contexto**; **private** BaseDatos **baseDatos**; **public** ActivityTestRule<BusquedaDestino> **reglaActivity**; **private** BusquedaDestino **mngActivity**; **private** ArrayList<Ruta> **lstRutas**;

	Función:
	
	@Before
	public void setUp() **throws** Exception {
	reglaActivity = **new** ActivityTestRule<>(BusquedaDestino.**class**);
	reglaActivity.launchActivity(**null**);
	mngActivity = **reglaActivity**.getActivity();
	reglaActivity = **new** ActivityTestRule<>(BusquedaDestino.**class**);
	contexto= InstrumentationRegistry.*getTargetContext*();
	baseDatos=**new** BaseDatos(**contexto**);
	}

	Métodos Colaboradores

	public void llenarRutas()
	{
	lstRutas= **new** ArrayList<>();
	lstRutas.add(**new** Ruta(1,**"quito"**,**"guayaquil"**,1));
	lstRutas.add(**new** Ruta(2,**"riobamba"**,**"cuenca"**,11));
	lstRutas.add(**new** Ruta(3,**"macas"**,**"ambato"**,13));
	lstRutas.add(**new** Ruta(4,**"guayaquil"**,**"loja"**,7));
	}

	Post Ejecución

	Función:
	
	@After
	public void tearDown() **throws** Exception {
	baseDatos.close();
	}

Realizado por: Wilmer B. 2018.

Caso de prueba 01 de historia de usuario 08.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 05	**Funcionalidad:** HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino
Código: TC_01	**Descripción:** Cargar lista ciudades origen desde base de datos local
Entrada(s)	

```
@Test
public void cargarListaCiudadesOrigen() throws Exception {


   String[] resp=baseDatos.obtenerCiudadesOrigen();
   assertNotNull(resp);
}
```

Resultado Esperado	
Cuando se manda a cargar la lista de ciudades origen de la base de datos, está lista se devolverá como resultado en una lista de string la misma que tendrá un valor diferente de **null**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 105ms

Realizado por: Wilmer B. 2018.

Caso de prueba 02 de historia de usuario 08.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 05	**Funcionalidad:** HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino
Código: TC_02	**Descripción:** Chequear cuando no hay lista de ciudades origen desde base de datos local

Entrada(s)

```
@Test
public void noExistenCiudadesOrigen() throws Exception {
   SQLiteDatabase db=baseDatos.getWritableDatabase();
   baseDatos.eliminarContenidoTabla(db,"DELETE FROM RUTA");
   String[] resp=baseDatos.obtenerCiudadesOrigen();
   assertNull(resp);
}
```

Resultado Esperado	
Al intentar cargar lista de ciudades origen desde la base de datos, y si estas no han sido registradas, entonces se devolverá como resultado una lista de strings diferente de **null**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 209ms

Realizado por: Wilmer B. 2018.

Caso de prueba 03 de historia de usuario 08.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 05	**Funcionalidad:** HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino
Código: TC_03	**Descripción:** Cargar lista ciudades destino desde la base de datos local

Entrada(s)

```
@Test
public void cargarListaCiudadesDestinoBasadoCiudadOrigen() throws Exception {
    ArrayList<Ruta> resp=baseDatos.obtenerCiudadesDestino("guayaquil");
    assertEquals(true,resp.size()>0);
}
```

Resultado Esperado

Cuando se manda a cargar la lista de ciudades destino que tiene una ciudad origen, entonces se devolverá estás ciudades en una lista de rutas la cual tendrá un número de elementos mayor que 0

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 285ms

Realizado por: Wilmer B. 2018.

Caso de prueba 04 de historia de usuario 08.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 05	**Funcionalidad:** HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino
Código: TC_04	**Descripción:** Chequear cuando la lista ciudades destino está vacía

Entrada(s)

```
@Test
public void noExistenCiudadesDestinoBasadoCiudadOrigen() throws Exception {
    ArrayList<Ruta> resp=baseDatos.obtenerCiudadesDestino("salcedo");
    assertEquals(0,resp.size());
}
```

Resultado Esperado

Al intentar extraer la lista de ciudades destino que tiene una ciudad origen, y no existen dichas ciudades destino, entonces devolverá una lista de rutas con un número de elementos igual a 0

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 155ms

Realizado por: Wilmer B. 2018.

Caso de prueba 05 de historia de usuario 08.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 05	**Funcionalidad:** HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino
Código: TC_05	**Descripción:** Extraer lista de Strings de ciudades destino de una lista de objetos de ruta

Entrada(s)

```
@Test
public void convertirListaObjsRutasToString() {
   llenarRutas();
   String[] lstCiudades={"Guayaquil","Cuenca","Ambato","Loja"};
   String [] respuesta= mngActivity.convertirLstObjRutasCadena(lstRutas);
   assertArrayEquals(lstCiudades,respuesta);
}
```

Resultado Esperado

Para obtener la lista de ciudades destino de una lista de rutas, estas se transforman en una lista de strings, en este caso será el resultado

```
String[] lstCiudades={"Guayaquil","Cuenca","Ambato","Loja"};
```

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 984ms

Realizado por: Wilmer B. 2018.

Caso de prueba 06 de historia de usuario 08.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 05	**Funcionalidad:** HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino
Código: TC_06	**Descripción:** Formatear cadena de texto a formato donde las palabras empiezan con letra mayúscula

Entrada(s)

```
@Test
public void formatearCadenaTexto() {
   String texto="EstO es UNA pRuebA";
   String respuesta= ConvTexto.primerasLetrasMayuscula(texto);
   String esperando="Esto Es Una Prueba";
   assertEquals(esperando,respuesta);
}
```

Resultado Esperado

Cuando se manda a formatear una cadena de texto para ser mostrada posteriormente, entonces devolverá como resultado dicha cadena con cada nueva palabra empezando con letra mayúscula y las demás minúsculas, para este caso la respuesta es: String esperando=**"Esto Es Una Prueba"**;

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 331ms

Realizado por: Wilmer B. 2018.

Caso de prueba 07 de historia de usuario 08.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 05	**Funcionalidad:** HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino
Código: TC_07	**Descripción:** Obtener lista de horarios en base a una ruta específica

Entrada(s)

```
@Test
public void cargarHorariosBasadoRuta() {
   ArrayList<InformacionHorarios> resp=baseDatos.obtenerHorarios(9);
   assertEquals(true,resp.size()>0);
}
```

Resultado Esperado

Cuando se carga los horarios basados en el id de una ruta, como resultado se tendrá una lista de Información de horarios, la misma que tendrá un número de elementos mayor a 0

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 804ms

Realizado por: Wilmer B. 2018.

Caso de prueba 08 de historia de usuario 08.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 05	**Funcionalidad:** HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino
Código: TC_08	**Descripción:** Chequear que la lista de horarios en base a una ruta específica este vacía

Entrada(s)

```
@Test
public void noExisteHorariosBasadoRuta() {
   ArrayList<InformacionHorarios> resp=baseDatos.obtenerHorarios(55);
   assertEquals(0,resp.size());
}
```

Resultado Esperado	
Cuando se carga los horarios basados en el id de una ruta y no existen horarios, como resultado se tendrá una lista de Información de horarios, la misma que tendrá un número de elementos igual a 0	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 331ms

Realizado por: Wilmer B. 2018.

Caso de prueba 09 de historia de usuario 08.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 05	**Funcionalidad:** HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino
Código: TC_09	**Descripción:** Obtener id de una ruta

Entrada(s)

```
@Test
public void obtenerIdRuta() {
   llenarRutas();
   int respuesta=mngActivity.obtenerIdRuta("riobamba","cuenca",lstRutas);
   assertEquals(2,respuesta);
}
```

Resultado Esperado	
Cuando se obtiene el id de una ruta basado en la ciudad origen y destino, dentro de una lista de rutas, se obtiene como resultado un número entero que representa al id, en este caso es: 2	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 829ms

Realizado por: Wilmer B. 2018.

Caso de prueba 10 de historia de usuario 08.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 05	**Funcionalidad:** HU_08 Implementación de funcionalidad para búsqueda de horarios en base a ciudad origen y destino
Código: TC_10	**Descripción:** Chequear cuando no se ha obtenido id de ruta

Entrada(s)

```
@Test
public void noExisteIdRuta() {
   llenarRutas();
   int respuesta=mngActivity.obtenerIdRuta("riobamba","loja",lstRutas);
```

assertEquals(0,respuesta); }
Resultado Esperado
Cuando se intenta obtener el id de una ruta basado en la ciudad origen y destino, dentro de una lista de rutas, se obtiene como resultado un número entero con valor 0 el cual representa significa que no se encontró id

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 31ms

Realizado por: Wilmer B. 2018.

Tabla de resultados en pruebas automáticas del sprint 5 historia de usuario 08.

RESULTADO	
Cantidad de pruebas	10
Tiempo total en ejecución de pruebas Sprint 5, HU_08	11s 64ms

Realizado por: Wilmer B. 2018.

PRUEBAS AUTOMÁTICAS SPRINT 5-HISTORIA DE USUARIO 09

Tabla de requisitos en sprint 5 historia de usuario 09.

REQUISITOS DE TEST SPRINT 5, HU_09	**Pre-Ejecución**

	Variables:
	public ActivityTestRule<BusquedaDestino> **reglaActivity**;
	private BusquedaDestino **mngActivity**;
	private ArrayList<InformacionHorarios> **lstInf**;
	
	Función:
	
	@Before
	public void setUp() **throws** Exception {
	reglaActivity = **new** ActivityTestRule<>(BusquedaDestino.**class**);
	reglaActivity.launchActivity(**null**);
	mngActivity = **reglaActivity**.getActivity();
	reglaActivity = **new** ActivityTestRule<>(BusquedaDestino.**class**);
	}

	Métodos Colaboradores
	public void llenarListaInfHorarios() { **lstInf**= **new** ArrayList<>();

```
    String[][] lstHoras= new String[3][2];
    lstHoras[0][0]="11:30:00";
    lstHoras[0][1]="1";

    lstHoras[1][0]="04:05:00";
    lstHoras[1][1]="2";

    lstHoras[2][0]="09:12:00";
    lstHoras[2][1]="3";
    lstInf.add(new InformacionHorarios("patria",1,1234,lstHoras,1));
}
```

Post Ejecución

Sin métodos post ejecución

Realizado por: Wilmer B. 2018.

Caso de prueba 01 de historia de usuario 09.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 05	**Funcionalidad:** HU_09 Implementación de funcionalidad para obtener la próxima salida en base a la hora actual
Código: TC_01	**Descripción:** Adaptar lista de información de horarios, rutas y transportes para presentación en interfaz

Entrada(s)

```
@Test
public void adaptarListaInfoHorariosToPresentacion() {
    llenarListaInfHorarios();
    ArrayList<AdaptadorInformacion> lstResp;
    lstResp=mngActivity.adaptarObjetosParaPresentacion(lstInf);
    assertEquals(true,lstResp.size()>0);
}
```

Resultado Esperado

Cuando se necesita cargar una lista de horarios y adaptarla para posterior presentación este proceso, devolverá como resultado una lista con un número mayor a 0

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 908ms

Realizado por: Wilmer B. 2018.

Caso de prueba 02 de historia de usuario 09.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 05	**Funcionalidad:** HU_09 Implementación de funcionalidad para obtener la próxima salida en base a la hora actual
Código: TC_02	**Descripción:** Ordenar ascendente la lista de horarios de una ruta específica adaptada

Entrada(s)

```
@Test
public void ordenarHorariosRutaAscendentemente() {
   llenarListaInfHorarios();
   ArrayList<AdaptadorInformacion> lstAdap;
   lstAdap=mngActivity.adaptarObjetosParaPresentacion(lstInf);
   ArrayList<AdaptadorInformacion> lstResp;
   lstResp= ManejoHoras.ordenarListaHorarios(lstAdap);
   assertEquals("04:05:00",lstResp.get(0).getHora_salida());
   assertEquals("09:12:00",lstResp.get(1).getHora_salida());
   assertEquals("11:30:00",lstResp.get(2).getHora_salida());
}
```

Resultado Esperado

Cuando se manda a ordenar una lista de horarios, entonces como resultado obtendremos dicha lista ordenada ascendentemente, en este caso tenemos como respuesta:

1.-"04:05:00"

2.-"09:12:00"

3.-"11:30:00"

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 131ms

Realizado por: Wilmer B. 2018.

Caso de prueba 03 de historia de usuario 09.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 05	**Funcionalidad:** HU_09 Implementación de funcionalidad para obtener la próxima salida en base a la hora actual
Código: TC_03	**Descripción:** Verificar la hora actual es AM

Entrada(s)

```
@Test
public void verificarHoraAM() {
   int Respuesta=ManejoHoras.verificarTiempoAmPm();
```

assertEquals(0,Respuesta); }
Resultado Esperado
Para verificar que una hora específica sea am, al realizar este proceso devolverá como resultado un valor 0 que significa AM
Evaluación de la prueba: Exitosa | **Tiempo de ejecución:** 1s 761ms

Realizado por: Wilmer B. 2018.

Caso de prueba 04 de historia de usuario 09.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 05	**Funcionalidad:** HU_09 Implementación de funcionalidad para obtener la próxima salida en base a la hora actual
Código: TC_04	**Descripción:** Verificar la hora actual es PM
Entrada(s)	

@Test
public void verificarHoraPM() {
int Respuesta=ManejoHoras.*verificarTiempoAmPm*();
assertEquals(1,Respuesta);
}	

Resultado Esperado	
Para verificar que una hora específica sea pm, al realizar este proceso devolverá como resultado un valor 1 que significa PM	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 330ms

Realizado por: Wilmer B. 2018.

Caso de prueba 05 de historia de usuario 09.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 05	**Funcionalidad:** HU_09 Implementación de funcionalidad para obtener la próxima salida en base a la hora actual
Código: TC_05	**Descripción:** Obtener el día de la semana actual numéricamente
Entrada(s)	
@Test **public void** obtenerDiaActualNumerico() { Calendar cal=Calendar.*getInstance*(); **int** diaAux=cal.get(Calendar.***DAY_OF_WEEK***);	

`int d= (diaAux==1)?7:diaAux-1;`
 	
`int dia=ManejoHoras.obtenerDiaDeLaSemanaNumero();`
`assertEquals(d,dia);`
}

Resultado Esperado
Para mandar a cargar el día actual de forma numérica, este proceso devolverá como resultado el día actual, correspondiendo a 1 como día lunes y a 7 como día domingo,
Evaluación de la prueba: Exitosa | **Tiempo de ejecución:** 1s 310ms

Realizado por: Wilmer B. 2018.

Caso de prueba 06 de historia de usuario 09.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 05	**Funcionalidad:** HU_09 Implementación de funcionalidad para obtener la próxima salida en base a la hora actual
Código: TC_06	**Descripción:** Convertir día numérico actual a día actual escrito
Entrada(s)	

`@Test`
`public void convertirDiaNumericoToTexto() {`
`String dia=ManejoHoras.convertirDiaNumeroToTexto(6);`
`assertEquals("Sábado",dia);`
}	

Resultado Esperado	
Cuando se necesita convertir el número de un día a texto, se obtendrá como resultado dicho día de manera escrita, en este caso se obtiene el resultado **"Sábado"**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 156ms

Realizado por: Wilmer B. 2018.

Caso de prueba 07 de historia de usuario 09.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 05	**Funcionalidad:** HU_09 Implementación de funcionalidad para obtener la próxima salida en base a la hora actual
Código: TC_07	**Descripción:** Obtener hora actual normal
Entrada(s)	

@Test
public void obtenerHoraActualSistema() {
String dia=ManejoHoras.*obtenerHoraActual*();
assertEquals(**"23:30:00 PM"**,dia);
}

Resultado Esperado
Cuando se necesita obtener la hora actual del sistema, este proceso arrojará como resultado una variable string con formato **"23:30:00 PM"**
Evaluación de la prueba: Exitosa | **Tiempo de ejecución:** 1s 154ms

Realizado por: Wilmer B. 2018.

Tabla de resultados en pruebas automáticas del sprint 5 historia de usuario 09.

RESULTADO	
Cantidad de pruebas	7
Tiempo total en ejecución de pruebas Sprint 5, HU_09	2s 550ms

Realizado por: Wilmer B. 2018.

Sprint 6

PRUEBAS AUTOMÁTICAS SPRINT 6-HISTORIA DE USUARIO 11

Tabla de requisitos en sprint 6 historia de usuario 11.

REQUISITOS DE TEST SPRINT 6, HU_11	**Pre-Ejecución**

	Variables:
	private HorariosTransporte **hrTransp**;
	private Context **contexto**;
	
	Función:
	
	@Before
	public void setUp() **throws** Exception {
	contexto=InstrumentationRegistry.*getTargetContext*();
	hrTransp =**new** HorariosTransporte(**contexto**);
	}

	Métodos Colaboradores
	No existen métodos colaboradores
	Post Ejecución

	Sin métodos post ejecución

Realizado por: Wilmer B. 2018.

Caso de prueba 01 de historia de usuario 11.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 06	**Funcionalidad:** HU_11 Implementación de funcionalidad para mostrar horarios por días
Código: TC_01	**Descripción:** Convertir día numérico a abreviación de día
Entrada(s)	

@Test
public void convertirDiaNumericoToAbreviacionTexto() {
String dia=**hrTransp**.formarCadenaDias(6);
assertEquals(**"Sáb|"**,dia);
}	

Resultado Esperado	
El proceso de transformar un día numérico a abreviación, el resultado de este proceso dará como resultado la abreviación de ese día textualmente con las 3 primeras letras de dicho día, en este caso: **"Sáb|"**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 02 de historia de usuario 11.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 06	**Funcionalidad:** HU_11 Implementación de funcionalidad para mostrar horarios por días
Código: TC_02	**Descripción:** Concatenación de días numéricos a días escritos laborables
Entrada(s)	

@Test
public void formarCadenaDiasLaborables() {
String dia=**hrTransp**.formarCadenaDias(12456);
assertEquals(**"Lun|Mar|Jue|Vie|Sáb|"**,dia);
}	

Resultado Esperado	

Para formar una cadena con los días laborables de manera abreviada, devolverá como resultado dicha cadena separando los días con un símbolo "|", en este caso es: **"Lun|Mar|Jue|Vie|Sáb|"**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Tabla de resultados en pruebas automáticas del sprint 6 historia de usuario 11.

RESULTADO	
Cantidad de pruebas	2
Tiempo total en ejecución de pruebas Sprint 6, HU_11	25ms

Realizado por: Wilmer B. 2018.

Sprint 7

PRUEBAS AUTOMÁTICAS SPRINT 7-HISTORIA DE USUARIO 12

Tabla de requisitos en sprint 7 historia de usuario 12.

REQUISITOS DE TEST SPRINT 7, HU_12	**Pre-Ejecución**

	Variables:
	`private Context context;`
	`private LocationListener locationListenerBest = new LocationListener(){`
	`@Override`
	`public void onLocationChanged(Location location) {}`
	`@Override`
	`public void onStatusChanged(String s, int i, Bundle bundle) {}`
	`@Override`
	`public void onProviderEnabled(String s) {}`
	`@Override`
	`public void onProviderDisabled(String s) {}`
	`};`
	
	Función:
	
	`@Before`
	`public void setUp() throws Exception {`
	`context=InstrumentationRegistry.getTargetContext();`
	`}`

	Métodos Colaboradores

	No existen métodos colaboradores
	Post Ejecución

	Función:
	
	@After
	public void tearDown() **throws** Exception {
	baseDatos.close();
	}

Realizado por: Wilmer B. 2018.

Caso de prueba 01 de historia de usuario 12.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android
Código: TC_01	**Descripción:** Activar servicio para localización LocationManager para uso de servicio de localización
Entrada(s)	

@Test
public void verificarActivacionServicioLocalizacion() {
LocalizacionGpsRed gpsRed=**new** LocalizacionGpsRed(**context**);
boolean respuesta= gpsRed.activarServicioLocationManager();
assertEquals(**true**,respuesta);
}	

Resultado Esperado	
Cuando se activa el servicio de localización, este proceso devolverá como resultado un valor positivo **true**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 02 de historia de usuario 12.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android
Código: TC_02	**Descripción:** Verificar proveedor de localización gps activo
Entrada(s)	

```
@Test
public void verificarLocalizacionGPSactiva() {
   Criteria c= new Criteria();
   LocationManager lm;
   c.setAccuracy(Criteria.ACCURACY_FINE);
   lm=(LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
   boolean respuesta= new LocalizacionGpsRed(context).localizacionGps(lm,c);
   assertTrue(respuesta);
}
```

Resultado Esperado	
Para verificar que la localización bajo el método del dispositivo gps esté activado, este proceso devolverá como resultado un valor verdadero **true**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 03 de historia de usuario 12.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android
Código: TC_03	**Descripción:** Verificar proveedor de localización gps inactivo

Entrada(s)

```
@Test
public void verificarLocalizacionGPSinActiva() throws Exception {
   Criteria c= new Criteria();
   c.setAccuracy(Criteria.ACCURACY_FINE);
   LocationManager lm;
   lm=(LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
   boolean respuesta= new LocalizacionGpsRed(context).localizacionGps(lm,c);
   assertFalse(respuesta);
}
```

Resultado Esperado	
Cuando el servicio de localización bajo el método del dispositivo gps esté desactivado, este proceso devolverá como resultado un valor negativo **false**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 04 de historia de usuario 12.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android
Código: TC_04	**Descripción:** Verificar proveedor de localización de red activo

Entrada(s)

```
@Test
public void verificarLocalizacionRedActiva() throws Exception {
   Criteria c= new Criteria();
   c.setAccuracy(Criteria.ACCURACY_FINE);
   LocationManager lm;
   lm=(LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
   boolean respuesta= new LocalizacionGpsRed(context).localizacionRed(lm,c);
   assertTrue(respuesta);
}
```

Resultado Esperado

Para verificar que la localización bajo el método de red esté activada, este proceso devolverá como resultado un valor verdadero **true**

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 25ms

Realizado por: Wilmer B. 2018.

Caso de prueba 05 de historia de usuario 12.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android
Código: TC_05	**Descripción:** Verificar proveedor de localización de red inactivo

Entrada(s)

```
@Test
public void verificarLocalizacionRedInActiva() throws Exception {
   Criteria c= new Criteria();
   c.setAccuracy(Criteria.ACCURACY_FINE);
   LocationManager lm;
   lm=(LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
   boolean respuesta= new LocalizacionGpsRed(context).localizacionRed(lm,c);
   assertFalse(respuesta);
}
```

Resultado Esperado	
Cuando el servicio de localización por red esté desactivado, este proceso devolverá como resultado un valor negativo **false**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 06 de historia de usuario 12.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android
Código: TC_06	**Descripción:** Omitir dato altitud en la localización

Entrada(s)

```
@Test
public void verificarOmicionAltitudGPS() throws Exception {
    Criteria c= new Criteria();
    c.setAccuracy(Criteria.ACCURACY_FINE);
    LocalizacionGpsRed gpsRed=new LocalizacionGpsRed(context);
    boolean respuesta= gpsRed.omitirAltitudDeLocalizacion(c);
    assertTrue(true);
}
```

Resultado Esperado	
Para omitir el dato de altitud de la localización por gps, este proceso devolverá un resultado positivo **true**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 07 de historia de usuario 12.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android
Código: TC_07	**Descripción:** Activar consumo de energía bajo

Entrada(s)

```
@Test
public void verificarActivacionConsumoBajoEnergia() throws Exception {
    Criteria c= new Criteria();
    c.setAccuracy(Criteria.ACCURACY_FINE);
```

```
    LocalizacionGpsRed gpsRed=new LocalizacionGpsRed(context);
    boolean respuesta= gpsRed.activarModoConsumoEnergiaBajo(c);
    assertTrue(respuesta);
}
```

Resultado Esperado	
Para activar el consumo de energía bajo, al usar la geolocalización, este proceso devolverá como resultado un valor positivo **true**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 08 de historia de usuario 12.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android
Código: TC_08	**Descripción:** Existe proveedor de localización activo

Entrada(s)

```
@Test
public void verificarProveedorLocalizacionActivo() throws Exception {
    Criteria c= new Criteria();
    c.setAccuracy(Criteria.ACCURACY_FINE);
    LocationManager lm;
    lm=(LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
    LocalizacionGpsRed gpsRed=new LocalizacionGpsRed(context);
    boolean respuesta= gpsRed.proveedorLocalizacionActivo(lm);
    assertTrue(respuesta);
}
```

Resultado Esperado	
Para verificar que algún proveedor de localización esté activado, este proceso devolverá como resultado un valor positivo **true**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 09 de historia de usuario 12.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android
Código: TC_09	**Descripción:** Chequear cuando no existe proveedor de localización activo

Entrada(s)

```
@Test
public void verificarProveedorLocalizacionInActivo() throws Exception {
    Criteria c= new Criteria();
    c.setAccuracy(Criteria.ACCURACY_FINE);
    LocationManager lm;
    lm=(LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
    LocalizacionGpsRed gpsRed=new LocalizacionGpsRed(context);
    boolean respuesta= gpsRed.proveedorLocalizacionActivo(lm);
    assertFalse(respuesta);
}
```

Resultado Esperado

Cuando ningún proveedor de localización esté activado, este proceso devolverá como resultado un valor negativo **false**

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 26ms

Realizado por: Wilmer B. 2018.

Caso de prueba 10 de historia de usuario 12.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android
Código: TC_10	**Descripción:** Desactivar servicio de localización, cuando ya no se necesita.

Entrada(s)

```
@Test
public void DesctivacionServicioLocalizacion() throws Exception {
    LocationManager lm;
    lm=(LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
    LocalizacionGpsRed gpsRed=new LocalizacionGpsRed(context);
    boolean respuesta= gpsRed.desactivarServicio(locationListenerBest,lm);
    assertEquals(true,respuesta);
}
```

Resultado Esperado	
Al desactivar el servicio de localización, este proceso devolverá como resultado un valor positivo **true** si la desactivación fue exitosa	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 11 de historia de usuario 12.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android
Código: TC_11	**Descripción:** Chequear que el mejor proveedor de localización no sea el de red

Entrada(s)

```
@Test
public void verificarMejorProveedorNoEsRed() throws Exception {
   Criteria c= new Criteria();
   c.setAccuracy(Criteria.ACCURACY_FINE);
   LocationManager lm;
   lm=(LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
   LocalizacionGpsRed gpsRed=new LocalizacionGpsRed(context);
   String respuesta= gpsRed.obtenerMejorProveedor(c,lm);
   assertNotEquals("network",respuesta);
}
```

Resultado Esperado	
Para verificar que el mejor proveedor de localización no es la red, este proceso devolverá un resultado diferente a **"network"**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 12 de historia de usuario 12.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android
Código: TC_12	**Descripción:** Verificar que el proveedor de localización esté en modo pasivo

Entrada(s)

```
@Test
public void verificarMejorProveedorEstePasivo() throws Exception {
   Criteria c= new Criteria();
   c.setAccuracy(Criteria.ACCURACY_FINE);
   LocationManager lm;
   lm=(LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
   LocalizacionGpsRed gpsRed=new LocalizacionGpsRed(context);
   String respuesta= gpsRed.obtenerMejorProveedor(c,lm);
   assertEquals("passive",respuesta);
}
```

Resultado Esperado	
Para verificar que el servicio de localización se encuentra modo pasivo, este proceso devolverá una cadena string con el valor **"passive"**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 13 de historia de usuario 12.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android
Código: TC_13	**Descripción:** Verificar que el mejor proveedor de localización sea el dispositivo gps

Entrada(s)

```
@Test
public void verificarMejorProveedorEsGPS() throws Exception {
   Criteria c= new Criteria();
   c.setAccuracy(Criteria.ACCURACY_FINE);
   LocationManager lm;
   lm=(LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
   LocalizacionGpsRed gpsRed=new LocalizacionGpsRed(context);
   String respuesta= gpsRed.obtenerMejorProveedor(c,lm);
   assertEquals("gps",respuesta);
}
```

Resultado Esperado
Para verificar que el mejor proveedor de localización es gps, este proceso devolverá como resultado un valor **"gps"**

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 14 de historia de usuario 12.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android
Código: TC_14	**Descripción:** Chequear que el mejor proveedor de localización no sea el dispositivo gps

Entrada(s)

```
@Test
public void verificarMejorProveedorNoEsGPS() throws Exception {
    Criteria c= new Criteria();
    c.setAccuracy(Criteria.ACCURACY_FINE);
    LocationManager lm;
    lm=(LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
    LocalizacionGpsRed gpsRed=new LocalizacionGpsRed(context);
    String respuesta= gpsRed.obtenerMejorProveedor(c,lm);
    assertNotEquals("gps",respuesta);
}
```

Resultado Esperado

Para verificar que el mejor proveedor de localización no sea el gps, este proceso devolverá como resultado un valor diferente a **"gps"**

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Caso de prueba 15 de historia de usuario 12.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_12 Implementación de funcionalidad para obtener coordenadas geográficas en Android
Código: TC_15	**Descripción:** Verificar que el proveedor de localización gps esté inactivo

Entrada(s)

```
@Test
public void verificarProveedorLocalizacionGPSinActivo() throws Exception {
    Criteria c= new Criteria();
    c.setAccuracy(Criteria.ACCURACY_FINE);
```

LocationManager lm;
lm=(LocationManager) **context**.getSystemService(Context.***LOCATION_SERVICE***);
boolean respuesta= **new** LocalizacionGpsRed(**context**).proveedorGpsActivo(lm);
assertFalse(respuesta);
}

Resultado Esperado
Para verificar que el proveedor de localización gps esté inactivo, este proceso devolverá como resultado un valor booleano igual a **false.**
Evaluación de la prueba: Exitosa | **Tiempo de ejecución:** 0ms

Realizado por: Wilmer B. 2018.

Tabla de resultados en pruebas automáticas del sprint 7 historia de usuario 12.

RESULTADO	
Cantidad de pruebas	15
Tiempo total en ejecución de pruebas Sprint 7, HU_12	76ms

Realizado por: Wilmer B. 2018.

PRUEBAS AUTOMÁTICAS SPRINT 7-HISTORIA DE USUARIO 13

Tabla de requisitos en sprint 7 historia de usuario 13.

REQUISITOS DE TEST SPRINT 7, HU_13	**Pre-Ejecución**
	Variables: **private** Context **context**= InstrumentationRegistry.*getTargetContext*(); **public** ActivityTestRule<DetalleRuta> **reglaActivity**; **private** DetalleRuta **mngActivity**;
	Métodos Colaboradores
	No existen métodos colaboradores
	Post Ejecución
	No existen métodos post ejecución

Realizado por: Wilmer B. 2018.

Caso de prueba 01 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_01	**Descripción:** Verificar si existe una conexión a internet activa

Entrada(s)

```
@Test
public void conexionActivaInternet() {
   ConnectivityManager cm;
   cm=(ConnectivityManager)
mngActivity.getSystemService(Context.CONNECTIVITY_SERVICE);
   NetworkInfo info = cm.getActiveNetworkInfo();
   assertTrue(mngActivity.hayConexionActiva(info));
}
```

Resultado Esperado

Para verificar que exista una conexión activa a internet, este proceso devolverá un valor **true** en caso de existir.

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 357ms

Realizado por: Wilmer B. 2018.

Caso de prueba 02 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_02	**Descripción:** Chequear que haya conexión activa de red wifi

Entrada(s)

```
@Test
public void conexionWiFiActiva() {
   ConnectivityManager cm;
   cm=(ConnectivityManager)
mngActivity.getSystemService(Context.CONNECTIVITY_SERVICE);
   NetworkInfo info = cm.getActiveNetworkInfo();
   assertEquals("wifi",mngActivity.obtenerTipoConexion(info));
}
```

Resultado Esperado

Para verificar que haya una conexión activa a través de wifi, este proceso devolverá como resultado un valor igual a **"wifi"** lo que significa que, si existe conexión activa a través de una red wifi	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 2s 932ms

Realizado por: Wilmer B. 2018.

Caso de prueba 03 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_03	**Descripción:** Chequear cuando no hay conexión activa de red wifi
Entrada(s)	

```
@Test
public void noHayConexionWiFiActiva() {
   ConnectivityManager cm;
   cm=(ConnectivityManager)
mngActivity.getSystemService(Context.CONNECTIVITY_SERVICE);
   NetworkInfo info = cm.getActiveNetworkInfo();
   assertNotEquals("wifi",mngActivity.obtenerTipoConexion(info));
}
```

Resultado Esperado	
Cuando no haya una conexión activa a través de wifi, este proceso devolverá como resultado un valor diferente a **"wifi"** lo que significa que, no existe conexión activa a través de wifi	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 683ms

Realizado por: Wilmer B. 2018.

Caso de prueba 04 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_04	**Descripción:** Verificar que haya conexión activa de datos móviles
Entrada(s)	

```
@Test
public void conexionMovilActiva() {
   ConnectivityManager cm;
   cm=(ConnectivityManager)
```

mngActivity.getSystemService(Context.***CONNECTIVITY_SERVICE***);
NetworkInfo info = cm.getActiveNetworkInfo();
assertEquals(**"mobile"**,**mngActivity**.obtenerTipoConexion(info));
}

Resultado Esperado
Para verificar que haya una conexión activa a través de datos móviles, este proceso devolverá como resultado un valor igual a **"mobile"** lo que significa que, si existe conexión activa a través de datos móviles

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 584ms

Realizado por: Wilmer B. 2018.

Caso de prueba 05 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_05	**Descripción:** Chequear cuando no hay conexión activa de datos móviles

Entrada(s)

@Test
public void noHayConexionMovilActiva() {
ConnectivityManager cm;
cm=(ConnectivityManager)
mngActivity.getSystemService(Context.***CONNECTIVITY_SERVICE***);
NetworkInfo info = cm.getActiveNetworkInfo();
assertNotEquals(**"mobile"**,**mngActivity**.obtenerTipoConexion(info));
}

Resultado Esperado
Cuando no haya una conexión activa a través de datos móviles, este proceso devolverá como resultado un valor diferente a **"mobile"** lo que significa que, no existe conexión activa a través de datos móviles

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 506ms

Realizado por: Wilmer B. 2018.

Caso de prueba 06 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_06	**Descripción:** Activar servicio para acceso a internet

Entrada(s)

```
@Test
public void ActivoServicioInternet() {
   ConnectivityManager cm;
   cm =(ConnectivityManager)
mngActivity.getSystemService(Context.CONNECTIVITY_SERVICE);
   NetworkInfo info = cm.getActiveNetworkInfo();
   assertTrue(mngActivity.activarServicioUsoInternet());
}
```

Resultado Esperado

Cuando se activa el servicio de acceso a internet, este proceso devuelve un valor positivo **true,** si se activó con éxito dicho servicio.

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 659ms

Realizado por: Wilmer B. 2018.

Caso de prueba 07 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_07	**Descripción:** Chequear cuando no existe servicio activo para acceso a internet

Entrada(s)

```
@Test
public void SinServicoActivoInternet() {
   ConnectivityManager cm;
   cm =(ConnectivityManager)
mngActivity.getSystemService(Context.CONNECTIVITY_SERVICE);
   NetworkInfo info = cm.getActiveNetworkInfo();
   assertEquals(null,mngActivity.obtenerTipoConexion(info));
}
```

Resultado Esperado

Para verificar cuando no se activa el servicio de acceso a internet, este proceso devuelve un valor **null**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 890ms

Realizado por: Wilmer B. 2018.

Caso de prueba 08 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_08	**Descripción:** Verificar si existe conectividad y disponibilidad a datos móviles

Entrada(s)

```
@Test
public void ExisteConectividad() {
   ConnectivityManager cm;
   cm =(ConnectivityManager)
mngActivity.getSystemService(Context.CONNECTIVITY_SERVICE);
   NetworkInfo info = cm.getActiveNetworkInfo();
   assertEquals(2,mngActivity.estadoServicioRed(info));
}
```

Resultado Esperado

Para verificar que exista conectividad a la red, este proceso devolverá como resultado un valor igual a 2 el cual significa que exista dicha conectividad

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 382ms

Realizado por: Wilmer B. 2018.

Caso de prueba 09 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_09	**Descripción:** Verificar si fue exitoso el abrir una petición http

Entrada(s)

```
@Test
public void abrirPeticionHttp() {
   String url="http://aplicacionesdanilo.pe.hu/ArchivosPHP/SincronizarDatosApp.php";
   boolean resp=new Peticion().abrirConexionHttp(url);
```

assertEquals(**true**,resp); }
Resultado Esperado
Al abrir una conexión con el servicio web que consumiremos, si esta acción es exitosa entonces se obtendrá un valor positivo **true**
Evaluación de la prueba: Exitosa — **Tiempo de ejecución:** 624ms

Realizado por: Wilmer B. 2018.

Caso de prueba 10 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_10	**Descripción:** Verificar estado de la petición sea aceptada
Entrada(s)	

@Test
public void estadoConexionAceptada() {
String url=**"http://aplicacionesdanilo.pe.hu/ArchivosPHP/SincronizarDatosApp.php"**;
HttpURLConnection con= Conexion.*ConectarConSW*(url);
boolean resp=**new** Peticion().peticionHttpAceptada(con);
assertTrue(resp);
}	

Resultado Esperado	
Para verificar el estado de la conexión, si fue aceptada, este proceso devolverá como respuesta un valor positivo **true**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 3s 305ms

Realizado por: Wilmer B. 2018.

Caso de prueba 11 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_11	**Descripción:** Chequear cuando la petición http fue rechazada
Entrada(s)	
@Test **public void** peticionHttpRechazada() { **boolean** resp=**new** Peticion().abrirConexionHttp(**null**);	

assertFalse(resp); }
Resultado Esperado
Para verificar el estado de la conexión, si fue rechazada, este proceso devolverá como respuesta un valor negativo **false**
Evaluación de la prueba: Exitosa | **Tiempo de ejecución:** 862ms

Realizado por: Wilmer B. 2018.

Caso de prueba 12 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_12	**Descripción:** Verificar resultado de petición http respuesta no esté vacía

Entrada(s)

```
@Test
public void existeResultadoPeticionHttp() {
    ArrayList<Parametro> lstParametros= new ArrayList<>();
    lstParametros.add(new Parametro("accion","sincronizar"));
    String url="http://aplicacionesdanilo.pe.hu/ArchivosPHP/SincronizarDatosApp.php";
    String respuesta=new Peticion().Enviar(url,lstParametros,context);
    assertNotNull(respuesta);
}
```

Resultado Esperado	
Para verificar que haya respuesta del servicio web que se consume, este proceso devolver una cadena de texto con la respuesta diferente de **null**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 19s 343ms

Realizado por: Wilmer B. 2018.

Caso de prueba 13 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_13	**Descripción:** Chequear cuando no hay resultado de consumo de servicio web

Entrada(s)

```
@Test
public void sinResultadoPeticionHttp() {
   ArrayList<Parametro> lstParametros= new ArrayList<>();
   lstParametros.add(new Parametro("accion",""));
   String url="http://aplicacionesdanilo.pe.hu/ArchivosPHP/SincronizarDatosApp.php";
   String respuesta=new Peticion().Enviar(url,lstParametros,context);
   assertEquals(respuesta,"");
}
```

Resultado Esperado	
Para verificar cuando no haya respuesta del servicio web que se consume, este proceso devolver una cadena de texto con la respuesta vacía `""`	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 987ms

Realizado por: Wilmer B. 2018.

Caso de prueba 14 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_14	**Descripción:** Verificar la disponibilidad de asientos de la respuesta traída de la petición http

Entrada(s)

```
@Test
public void ExistenDisponibilidadAsientos() {
   ArrayList<Parametro> lstParametros= new ArrayList<>();
   lstParametros.add(new Parametro("id_dpr","9"));
   lstParametros.add(new Parametro("id_hora","13"));
   lstParametros.add(new Parametro("accion","boletos_disponibles"));
   String url="http://aplicacionesdanilo.pe.hu/ArchivosPHP/SincronizarDatosApp.php";
   String respuesta=new Peticion().Enviar(url,lstParametros,context);
   assertNotNull(respuesta);
}
```

Resultado Esperado	
Para verificar la disponibilidad de asientos online, consumiendo el servicio web, este proceso nos dará un resultado en forma de cadena, la cual será diferente de **null**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 3s 355ms

Realizado por: Wilmer B. 2018.

Caso de prueba 15 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_15	**Descripción:** Verificar que el "Resultado" a la consulta de boletos disponibles sea una consulta correcta "CC"

Entrada(s)

```
@Test
public void respuestaCorrectaBoletosDisponibles() {
   try {
      ArrayList<Parametro> lstParametros= new ArrayList<>();
      lstParametros.add(new Parametro("id_dpr","9"));
      lstParametros.add(new Parametro("id_hora","13"));
      lstParametros.add(new Parametro("accion","boletos_disponibles"));
      String
url="http://aplicacionesdanilo.pe.hu/ArchivosPHP/SincronizarDatosApp.php";
      String respuesta=new Peticion().Enviar(url,lstParametros,context);
      JSONObject resp=new JSONObject(respuesta);
      assertEquals("CC",resp.getString("Resultado"));
   } catch (JSONException e) {
      e.printStackTrace();
   }
}
```

Resultado Esperado

Para verificar el estado de consulta correcta, de consumo del servicio web de boletos disponibles, este proceso devolverá un estado **"CC"** como resultado, lo que significa que la consulta fue correcta

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 8s 184ms

Realizado por: Wilmer B. 2018.

Caso de prueba 16 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_16	**Descripción:** Verificar que el "Resultado" a la consulta de boletos disponibles sea una consulta incorrecta "CI"

Entrada(s)

@Test
public void respuestaIncorrectaBoletosDisponibles() {
try {
ArrayList<Parametro> lstParametros= **new** ArrayList<>();
lstParametros.add(**new** Parametro(**"id_dpr"**,**"0"**));
lstParametros.add(**new** Parametro(**"id_hora"**,**"23"**));
lstParametros.add(**new** Parametro(**"accion"**,**"boletos_disponibles"**));
String
url=**"http://aplicacionesdanilo.pe.hu/ArchivosPHP/SincronizarDatosApp.php"**;
String respuesta=**new** Peticion().Enviar(url,lstParametros,**context**);
JSONObject resp=**new** JSONObject(respuesta);
assertEquals(**"CI"**,resp.getString(**"Resultado"**));
} **catch** (JSONException e) {
e.printStackTrace();
}
}

Resultado Esperado
Para verificar cuando el estado de consulta es incorrecto, en el consumo del servicio web de boletos disponibles, este proceso devolverá un estado **"CI"** como resultado, lo que significa que la consulta fue incorrecta

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 6s 761ms

Realizado por: Wilmer B. 2018.

Caso de prueba 17 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_17	**Descripción:** Verificar el número de asientos disponibles
Entrada(s)	
@Test **public void** extraerAsientosDisponibles() { **int**[] lstAsientos={1,5,9,3,18,14}; **int**[] respEsperada={2,4,6,7,8,10,11,12,13,15,16,17,19,20}; **int** resp[]=**mngActivity**.numeroDeAsientosDisonibles(lstAsientos,20);	

assertArrayEquals(resp,respEsperada); }
Resultado Esperado
Para extraer los números de asientos disponibles de un bus, entonces este proceso devolverá como resultado una lista de enteros con los asientos disponibles, para este caso es: **int**[] respEsperada={2,4,6,7,8,10,11,12,13,15,16,17,19,20};

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 609ms

Realizado por: Wilmer B. 2018.

Caso de prueba 18 de historia de usuario 13.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 07	**Funcionalidad:** HU_13 Implementación de funcionalidad para consulta de boletos disponibles en base a horarios, ruta y transporte
Código: TC_18	**Descripción:** Verificar la cantidad de asientos ocupados

Entrada(s)

@Test
public void extraerAsientosNoDisponibles() {
int[] lstAsientos={1,5,9,3,18,14};
int resp=**mngActivity**.numeroDeAsientosOcupados(lstAsientos,20);
assertEquals(resp,14);
}

Resultado Esperado
Para verificar la cantidad de asientos ocupados de un bus, este proceso devolverá como resultado un número entero que indica esta cantidad, en este caso: 14

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 610ms

Realizado por: Wilmer B. 2018.

Tabla de resultados en pruebas automáticas del sprint 7 historia de usuario 13.

RESULTADO	
Cantidad de pruebas	18
Tiempo total en ejecución de pruebas Sprint 7, HU_13	55s 633ms

Realizado por: Wilmer B. 2018.

Sprint 8

PRUEBAS AUTOMÁTICAS SPRINT 8-HISTORIA DE USUARIO 14

Tabla de requisitos en sprint 8 historia de usuario 14.

REQUISITOS DE TEST SPRINT 8, HU_14

Pre-Ejecución

Variables:

```
private Context contexto;
private BaseDatos baseDatos;
public ActivityTestRule<MainActivity> reglaActivity;
private MainActivity mngActivity;
private ArrayList<Favorito> lstFavoritos;
```

Función:

```
@Before
public void setUp() throws Exception {
    reglaActivity = new ActivityTestRule<>(MainActivity.class);
    reglaActivity.launchActivity(null);
    mngActivity = reglaActivity.getActivity();
    reglaActivity = new ActivityTestRule<>(MainActivity.class);

    contexto=InstrumentationRegistry.getTargetContext();
    baseDatos=new BaseDatos(contexto);
}
```

Métodos Colaboradores

```
public void llenarListaFavoritos()
{
    lstFavoritos= new ArrayList<>();
    lstFavoritos.add(new Favorito(1,"quito","guayaquil","22:55"));
    lstFavoritos.add(new Favorito(2,"ambato","cuenca","18:00"));
    lstFavoritos.add(new Favorito(3,"riobamba","macas","12:00"));
    lstFavoritos.add(new Favorito(4,"puyo","baños","05:00"));
    lstFavoritos.add(new Favorito(5,"macas","guayaquil","04:55"));
}
```

Post Ejecución

	Función:
	
	@After
	public void tearDown() **throws** Exception {
	baseDatos.close();
	}

Realizado por: Wilmer B. 2018.

Caso de prueba 01 de historia de usuario 14.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 08	**Funcionalidad:** HU_14 Implementación de funcionalidad favoritos
Código: TC_01	**Descripción:** Cargar lista de rutas favoritas extraída desde base de datos local
Entrada(s)	

@Test
public void obtenerListaFavoritos() **throws** Exception {
 	
ArrayList<Favorito> resp=**baseDatos**.cargarFavoritos();
assertEquals(**true**,resp.size()>0);
}	

Resultado Esperado	
Cuando se carga la lista de rutas favoritas desde la base de datos, esta devolverá como resultado una lista con dichas rutas, la cual tendrá un número de elementos mayor a 0	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 456ms

Realizado por: Wilmer B. 2018.

Caso de prueba 02 de historia de usuario 14.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 08	**Funcionalidad:** HU_14 Implementación de funcionalidad favoritos
Código: TC_02	**Descripción:** Chuequear cuando no haya lista de favoritos extraída de la base de datos
Entrada(s)	
@Test **public void** noExisteListaFavoritos() **throws** Exception { SQLiteDatabase db=**baseDatos**.getWritableDatabase(); **baseDatos**.eliminarContenidoTabla(db,**"DELETE FROM FAVORITO"**);	

```
    ArrayList<Favorito> resp=baseDatos.cargarFavoritos();
    assertEquals(0,resp.size());
}
```

Resultado Esperado	
Cuando se intenta cargar la lista de rutas favoritas desde la base de datos, y aún no existen dichas rutas, esta devolverá como resultado una lista con dichas rutas, la cual tendrá un número de elementos igual a 0	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 456ms

Realizado por: Wilmer B. 2018.

Caso de prueba 03 de historia de usuario 14.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 08	**Funcionalidad:** HU_14 Implementación de funcionalidad favoritos
Código: TC_03	**Descripción:** Convertir lista de objeto de favoritos a lista de cadena string de favoritos

Entrada(s)

```
@Test
public void convertirLstObjFavoritosToListaString()
{
    llenarListaFavoritos();
    String[] resp= mngActivity.convertirListaObjetosFavoritosToString(lstFavoritos);
    String[] respEsp={"Quito/Guayaquil  22:55","Ambato/Cuenca
18:00","Riobamba/Macas  12:00","Puyo/Baños  05:00","Macas/Guayaquil  04:55"};
    assertArrayEquals(resp,respEsp);
}
```

Resultado Esperado	
Cuando se manda a transformar una lista de objetos favoritos a una cadena de strings, dicha cadena contendrá la ciudad origen y destino separadas por un símbolo “/” y al lado derecho el horario de dicha ruta, en este caso es: String[] respEsp={"Quito/Guayaquil 22:55","Ambato/Cuenca 18:00","Riobamba/Macas 12:00","Puyo/Baños 05:00","Macas/Guayaquil 04:55"};	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 181ms

Realizado por: Wilmer B. 2018.

Caso de prueba 04 de historia de usuario 14.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 08	**Funcionalidad:** HU_14 Implementación de funcionalidad favoritos
Código: TC_04	**Descripción:** Agregar ruta a favorito

Entrada(s)

```
@Test
public void insertarDatosTablaFAVORITO()
{
   Favorito fav= new Favorito();

   fav.setId_ruta(5);
   fav.setOrigen("riobamba");
   fav.setDestino("puyo");
   fav.setTransporte("patria");
   fav.setTiempo_estimado("02:55:00");
   fav.setTurno("normal");
   fav.setCosto(18.23);
   fav.setHora("06:23:00");
   fav.setFecha("12/02/2015");

   String resp=baseDatos.agregarFavorito(fav);
   assertEquals("insertado",resp);
}
```

Resultado Esperado

Cuando se manda a agregar una nueva ruta favorita a la base de datos y se inserta correctamente, se obtendrá un resultado en una cadena string con contenido **"insertado"**

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 307ms

Realizado por: Wilmer B. 2018.

Caso de prueba 05 de historia de usuario 14.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 08	**Funcionalidad:** HU_14 Implementación de funcionalidad favoritos
Código: TC_05	**Descripción:** Verificar si dicho favorito ya existe

Entrada(s)

```
@Test
public void yaExisteFavorito()
{
   Favorito fav= new Favorito();

   fav.setId_ruta(9);
   fav.setOrigen("riobamba");
   fav.setDestino("guayaquil");
   fav.setTransporte("riobamba");
   fav.setTiempo_estimado("00:00:00");
   fav.setTurno("normal");
   fav.setCcosto(0);
   fav.setHora("18:00:00");
   fav.setFecha("jueves, junio 8, 2017");

   boolean resp=baseDatos.existeFavorito(fav);
   assertTrue(true);
}
```

Resultado Esperado	
Cuando se intenta agregar una ruta favorita, pero dicha ruta ya existe registrada en la base de datos, se obtendrá una respuesta positiva con valor **true** que representa que dicha ruta favorita ya existe.	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 259ms

Realizado por: Wilmer B. 2018.

Caso de prueba 06 de historia de usuario 14.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 08	**Funcionalidad:** HU_14 Implementación de funcionalidad favoritos
Código: TC_06	**Descripción:** Obtener la ruta favorita a partir del id y la hora de la misma

Entrada(s)

```
@Test
public void cargarRutaFavorita() throws Exception {
   Favorito resp=baseDatos.obetenrRutaFavorita(1,"08:20:00");
   assertNotNull(resp);
}
```

Resultado Esperado

Cuando se manda a cargar una ruta específica en base al id y la hora de la misma, si la consulta es exitosa devolverá un objeto diferente de **null**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 456ms

Realizado por: Wilmer B. 2018.

Caso de prueba 07 de historia de usuario 14.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 08	**Funcionalidad:** HU_14 Implementación de funcionalidad favoritos
Código: TC_07	**Descripción:** Verificar si una ruta específica está registrada dentro de favorito
Entrada(s)	

```
@Test
public void noExistenRutaFavorita() throws Exception {
   Favorito resp=baseDatos.obetenrRutaFavorita(2,"13:10:00");
   assertNull(resp);
}
```

Resultado Esperado	
Cuando se intenta mandar a cargar una ruta específica en base al id y la hora de la misma, pero no existe dicha ruta favorita, la consulta devolverá un objeto igual a **null**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 281ms

Realizado por: Wilmer B. 2018.

Tabla de resultados en pruebas automáticas del sprint 8 historia de usuario 14.

RESULTADO	
Cantidad de pruebas	7
Tiempo total en ejecución de pruebas Sprint 8, HU_14	9s 274ms

Realizado por: Wilmer B. 2018.

PRUEBAS AUTOMÁTICAS SPRINT 8-HISTORIA DE USUARIO 15

Tabla de requisitos en sprint 8 historia de usuario 15.

REQUISITOS DE TEST SPRINT 8,	**Pre-Ejecución**
	Variables: **public** ActivityTestRule<DetalleRuta> **reglaActivity**; **private** DetalleRuta **mngActivity**; **private** BaseDatos **baseDatos**; **private** Context **contexto**;

	Función:
	
	@Before
	public void setUp() **throws** Exception {
	reglaActivity = **new** ActivityTestRule<>(DetalleRuta.**class**);
	reglaActivity.launchActivity(**null**);
	mngActivity = **reglaActivity**.getActivity();
	contexto =InstrumentationRegistry.*getTargetContext*();
	baseDatos=**new** BaseDatos(**contexto**);
	}

	Métodos Colaboradores
	No existen métodos colaboradores
	Post Ejecución

	Función:
	@After
	public void tearDown() **throws** Exception {
	if(**baseDatos** != **null**)
	baseDatos.close();
	}

Realizado por: Wilmer B. 2018.

Caso de prueba 01 de historia de usuario 15.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 08	**Funcionalidad:** HU_15 Implementación de funcionalidad llamada telefónica y cargar datos de agencia de cooperativa
Código: TC_01	**Descripción:** Obtener datos de la agencia
Entrada(s)	

@Test
public void cargarDependenciaEnBaseId() {
Dependencia resp=**baseDatos**.obtenerDependencia(1);
assertNotNull(resp);
}	

Resultado Esperado	
Al cargar una dependencia específica en base a su id, se obtendrá como resultado un objeto Dependencia diferente a **null**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 805ms

Caso de prueba 02 de historia de usuario 15.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 08	**Funcionalidad:** HU_15 Implementación de funcionalidad llamada telefónica y cargar datos de agencia de cooperativa
Código: TC_02	**Descripción:** Chequear cuando no existe datos de la agencia
Entrada(s)	

@Test
public void noExisteDependenciaEnBaseId() {
Dependencia resp=**baseDatos**.obtenerDependencia(2);
assertNull(resp);
}	

Resultado Esperado	
Al intentar cargar una dependencia específica en base a su id, y la misma aún no ha sido registrada, se obtendrá como resultado un objeto Dependencia igual a **null**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 503ms

Realizado por: Wilmer B. 2018.

Caso de prueba 03 de historia de usuario 15.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 08	**Funcionalidad:** HU_15 Implementación de funcionalidad llamada telefónica y cargar datos de agencia de cooperativa
Código: TC_03	**Descripción:** Verificar permiso de llamada activo
Entrada(s)	

@Test
public void permisoConcedidoLlamar() {
boolean resp=**mngActivity**.llamarDependencia(**null**,**mngActivity**);
assertTrue(resp);
}	

Resultado Esperado	
Para verificar que exista el permiso de llamada a un número telefónico, este proceso devolverá una respuesta positiva **true** la cual significa que existe el permiso de dicho servicio	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 603ms

Realizado por: Wilmer B. 2018.

Caso de prueba 04 de historia de usuario 15.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 08	**Funcionalidad:** HU_15 Implementación de funcionalidad llamada telefónica y cargar datos de agencia de cooperativa
Código: TC_04	**Descripción:** Chequear cuando no existe permiso de llamada activo
Entrada(s)	

```
@Test
public void permisoDenegadoLlamar() {
   boolean resp=mngActivity.llamarDependencia(null,mngActivity);
   assertFalse(resp);
}
```

Resultado Esperado	
Para verificar cuando no existe el permiso de llamada a un número telefónico, este proceso devolverá una respuesta negativa **false** la cual significa que no existe el permiso de dicho servicio	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 442ms

Realizado por: Wilmer B. 2018.

Caso de prueba 05 de historia de usuario 15.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 08	**Funcionalidad:** HU_15 Implementación de funcionalidad llamada telefónica y cargar datos de agencia de cooperativa
Código: TC_05	**Descripción:** Convertir número string a formato de número para llamar
Entrada(s)	

```
@Test
public void numeroTelefonicoFormateadoValido() {
   Uri respEsperada=Uri.parse("tel:"+"03204596");
   Uri resp=mngActivity.convertirNumeroFormatoLLamar("03204596");
   assertEquals(resp,respEsperada);
}
```

Resultado Esperado	
Para formatear una cadena con número telefónico a un número valido para llamar, este proceso devolverá una respuesta con un objeto de tipo Uri	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 579ms

Realizado por: Wilmer B. 2018.

Tabla de resultados en pruebas automáticas del sprint 8 historia de usuario 15.

RESULTADO	
Cantidad de pruebas	5
Tiempo total en ejecución de pruebas Sprint 8, HU_15	3s 932ms

Realizado por: Wilmer B. 2018.

Sprint 9

PRUEBAS AUTOMÁTICAS SPRINT 9-HISTORIA DE USUARIO 17

Tabla de requisitos en sprint 9 historia de usuario 17.

REQUISITOS DE TEST SPRINT 9, HU_17	**Pre-Ejecución**

	Variables:
	private Context **contexto**;
	private BaseDatos **baseDatos**;
	public ActivityTestRule<Configuracion> **reglaActivity**;
	private Configuracion **mngActivity**;
	
	Función:
	
	@Before
	public void setUp() **throws** Exception {
	reglaActivity = **new** ActivityTestRule<>(Configuracion.**class**);
	reglaActivity.launchActivity(**null**);
	mngActivity = **reglaActivity**.getActivity();
	reglaActivity = **new** ActivityTestRule<>(Configuracion.**class**);
	
	contexto= InstrumentationRegistry.*getTargetContext*();
	baseDatos=**new** BaseDatos(**contexto**);
	}

	Métodos Colaboradores
	No existen métodos colaboradores
	Post Ejecución
	Función:

	@After
	public void tearDown() **throws** Exception {
	baseDatos.close();
	}

Realizado por: Wilmer B. 2018.

Caso de prueba 01 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_01	**Descripción:** Activar servicio de wifi
Entrada(s)	

@Test
public void activarServicioAccesoRedWifi() {
SincronizarBasesDatos sbd=**new** SincronizarBasesDatos(**mngActivity**,**null**);
boolean resp=sbd.activarServicioAccesoRed();
assertTrue(resp);
}	

Resultado Esperado	
Para activar el servicio de acceso a la red, este proceso devolverá como resultado un valor positivo **true**, si dicho servicio fue activado	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 383ms

Realizado por: Wilmer B. 2018.

Caso de prueba 02 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_02	**Descripción:** Verificar conexión a la red activa
Entrada(s)	
@Test **public void** verificarConexionActivaRed() { ConnectivityManager cm; cm= (ConnectivityManager) **mngActivity**.getSystemService(Context.***CONNECTIVITY_SERVICE***); NetworkInfo info = cm.getActiveNetworkInfo();	

```
    SincronizarBasesDatos sbd=new SincronizarBasesDatos(mngActivity,null);
    boolean resp=sbd.noExisteConexionARed(info);
    assertFalse(resp);
}
```

Resultado Esperado	
Para verificar que existe una conexión de red activa, este proceso devolverá como resultado un valor igual a **false** lo que significa que existe una conexión activa	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 582ms

Realizado por: Wilmer B. 2018.

Caso de prueba 03 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_03	**Descripción:** Chequear cuando no hay conexión activa hacia la red

Entrada(s)

```
@Test
public void verificarSinConexionActivaRed() {
    ConnectivityManager cm;
    cm= (ConnectivityManager)
mngActivity.getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo info = cm.getActiveNetworkInfo();
    SincronizarBasesDatos sbd=new SincronizarBasesDatos(mngActivity,null);
    boolean resp=sbd.noExisteConexionARed(info);
    assertTrue(resp);
}
```

Resultado Esperado	
Para verificar que cuando no existe una conexión de red activa, este proceso devolverá como resultado un valor igual a **true** lo que significa que no existe una conexión activa	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 583ms

Realizado por: Wilmer B. 2018.

Caso de prueba 04 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_04	**Descripción:** Verificar que el servicio de conexión a la red esté disponible

Entrada(s)

```
@Test
public void verificarConexionDisponibles() {
   ConnectivityManager cm;
   cm= (ConnectivityManager)
mngActivity.getSystemService(Context.CONNECTIVITY_SERVICE);
   NetworkInfo info = cm.getActiveNetworkInfo();
   SincronizarBasesDatos sbd=new SincronizarBasesDatos(mngActivity,null);
   boolean resp=sbd.disponibilidadRed(info);
   assertTrue(resp);
}
```

Resultado Esperado

Para verificar que la conexión a la red esté disponible, este proceso devolverá un valor positivo **true**

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 713ms

Realizado por: Wilmer B. 2018.

Caso de prueba 05 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_05	**Descripción:** Verificar que el tipo de conexión sea wifi

Entrada(s)

```
@Test
public void verificarConexionActivaWifi() {
   ConnectivityManager cm;
   cm= (ConnectivityManager)
mngActivity.getSystemService(Context.CONNECTIVITY_SERVICE);
   NetworkInfo info = cm.getActiveNetworkInfo();
   SincronizarBasesDatos sbd=new SincronizarBasesDatos(mngActivity,null);
   String resp=sbd.obtenerTipoConexion(info);
```

assertEquals("**wifi**",resp); }
Resultado Esperado
Para verificar que el tipo de conexión a la red sea a través de wifi, este proceso devolverá un resultado con valor **"wifi"**
Evaluación de la prueba: Exitosa — **Tiempo de ejecución:** 2s 3ms

Realizado por: Wilmer B. 2018.

Caso de prueba 06 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_06	**Descripción:** Verificar el tipo de conexión de red sea datos móviles

Entrada(s)

```
@Test
public void verificarConexionActivaMovil() {
   ConnectivityManager cm;
   cm= (ConnectivityManager)
mngActivity.getSystemService(Context.CONNECTIVITY_SERVICE);
   NetworkInfo info = cm.getActiveNetworkInfo();
   SincronizarBasesDatos sbd=new SincronizarBasesDatos(mngActivity,null);
   String resp=sbd.obtenerTipoConexion(info);
   assertEquals("mobile",resp);
}
```

Resultado Esperado	
Para verificar que el tipo de conexión a la red sea a través de datos móviles, este proceso devolverá un resultado con valor **"mobile"**	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 284ms

Realizado por: Wilmer B. 2018.

Caso de prueba 07 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_07	**Descripción:** Verificar que la respuesta del servicio web traiga un "Resultado" con "CC" de consulta correcta

Entrada(s)	

@Test
public void verificarConsumoSWConResultado() {
try {
ArrayList<Parametro> lstParametros= **new** ArrayList<>();
lstParametros.add(**new** Parametro(**"accion"**,**"sincronizar"**));
String url=**"http://aplicacionesdanilo.pe.hu/ArchivosPHP/SincronizarDatosApp.php"**;
String respuesta=**new** Peticion().Enviar(url,lstParametros,**contexto**);
JSONObject resp=**new** JSONObject(respuesta);
assertEquals(**"CC"**,resp.getString(**"Resultado"**));
} **catch** (JSONException e) {
e.printStackTrace();
}
}	

Resultado Esperado	
Para verificar que el consumo del servicio web de sincronización traiga resultados, este proceso devolverá como resultado un valor de estado **"CC"** lo que significa una consulta correcta	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 813ms

Realizado por: Wilmer B. 2018.

Caso de prueba 08 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_08	**Descripción:** Verificar que la respuesta del servicio web traiga un "Resultado" con "CI" de consulta incorrecta
Entrada(s)	

@Test
public void verificarConsumoSWSinResultado() {
try {
ArrayList<Parametro> lstParametros= **new** ArrayList<>();
lstParametros.add(**new** Parametro(**""**,**"sincronizar"**));
String url=**"http://aplicacionesdanilo.pe.hu/ArchivosPHP/SincronizarDatosApp.php"**;
String respuesta=**new** Peticion().Enviar(url,lstParametros,**contexto**);	

```
        JSONObject resp=new JSONObject(respuesta);
        assertEquals("CI",resp.getString("Resultado"));
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

Resultado Esperado

Cuando el consumo del servicio web de sincronización no traiga resultados, este proceso devolverá como resultado un valor de estado **"CI"** lo que significa una consulta incorrecta

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 2s 190ms

Realizado por: Wilmer B. 2018.

Caso de prueba 09 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_09	**Descripción:** Verificar que existan datos de la tabla “COOPERATIVA”

Entrada(s)

```
@Test
public void verificarContenidoTablaCooperativa() {
    int respuesta=baseDatos.verificarContenidoTabla("COOPERATIVA");
    assertEquals(true,respuesta > 0);
}
```

Resultado Esperado

Para verificar que exista contenido en la tabla **COOPERATIVA**, este proceso devolverá un valor entero, el cual representa la cantidad de registros, el mismo que tiene que ser mayor a 0 si existen registros en dicha tabla

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 408ms

Realizado por: Wilmer B. 2018.

Caso de prueba 10 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_10	**Descripción:** Verificar que existan datos de la tabla “DEPENDENCIA”

Entrada(s)

```
@Test
public void verificarContenidoTablaDependencia() {
   int respuesta=baseDatos.verificarContenidoTabla("DEPENDENCIA");
   assertEquals(true,respuesta > 0);
}
```

Resultado Esperado	
Para verificar que exista contenido en la tabla **DEPENDENCIA**, este proceso devolverá un valor entero, el cual representa la cantidad de registros, el mismo que tiene que ser mayor a 0 si existen registros en dicha tabla	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 383ms

Realizado por: Wilmer B. 2018.

Caso de prueba 11 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_11	**Descripción:** Verificar que existan datos de la tabla “TERMINAL”

Entrada(s)

```
@Test
public void verificarContenidoTablaTerminal() {
   int respuesta=baseDatos.verificarContenidoTabla("TERMINAL");
   assertEquals(true,respuesta > 0);
}
```

Resultado Esperado	
Para verificar que exista contenido en la tabla **TERMINAL**, este proceso devolverá un valor entero, el cual representa la cantidad de registros, el mismo que tiene que ser mayor a 0 si existen registros en dicha tabla	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 232ms

Realizado por: Wilmer B. 2018.

Caso de prueba 12 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_12	**Descripción:** Verificar que existan datos de la tabla “RUTA”

Entrada(s)

```
@Test
public void verificarContenidoTablaRuta() {
   int respuesta=baseDatos.verificarContenidoTabla("RUTA");
   assertEquals(true,respuesta > 0);
}
```

Resultado Esperado	
Para verificar que exista contenido en la tabla **RUTA**, este proceso devolverá un valor entero, el cual representa la cantidad de registros, el mismo que tiene que ser mayor a 0 si existen registros en dicha tabla	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 911ms

Realizado por: Wilmer B. 2018.

Caso de prueba 13 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_13	**Descripción:** Verificar que existan datos de la tabla “HORARIO”

Entrada(s)

```
@Test
public void verificarContenidoTablaHorario() {
   int respuesta=baseDatos.verificarContenidoTabla("HORARIO");
   assertEquals(true,respuesta > 0);
}
```

Resultado Esperado	
Para verificar que exista contenido en la tabla **HORARIO**, este proceso devolverá un valor entero, el cual representa la cantidad de registros, el mismo que tiene que ser mayor a 0 si existen registros en dicha tabla	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 708ms

Realizado por: Wilmer B. 2018.

Caso de prueba 14 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_14	**Descripción:** Verificar que existan datos de la tabla “TERM_PERTENECE_DEPEN”

Entrada(s)	

@Test
public void verificarContenidoTablaTerm_pertenece_depen() {
int respuesta=**baseDatos**.verificarContenidoTabla(**"TERM_PERTENECE_DEPEN"**);
assertEquals(**true**,respuesta > 0);
}	

Resultado Esperado	
Para verificar que exista contenido en la tabla **TERM_PERTENECE_DEPEN**, este proceso devolverá un valor entero, el cual representa la cantidad de registros, el mismo que tiene que ser mayor a 0 si existen registros en dicha tabla	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 532ms

Realizado por: Wilmer B. 2018.

Caso de prueba 15 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_15	**Descripción:** Verificar que existan datos de la tabla "DEPEN_POSEE_RUTAS"
Entrada(s)	

@Test
public void verificarContenidoTablaDepen_posee_ruta() {
int respuesta=**baseDatos**.verificarContenidoTabla(**"DEPEN_POSEE_RUTAS"**);
assertEquals(**true**,respuesta > 0);
}	

Resultado Esperado	
Para verificar que exista contenido en la tabla **DEPEN_POSEE_RUTAS**, este proceso devolverá un valor entero, el cual representa la cantidad de registros, el mismo que tiene que ser mayor a 0 si existen registros en dicha tabla	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 487ms

Realizado por: Wilmer B. 2018.

Caso de prueba 16 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_16	**Descripción:** Verificar que existan datos de la tabla “PROVINCIAS”
Entrada(s)	

```
@Test
public void verificarContenidoTablaProvincias() {
   int respuesta=baseDatos.verificarContenidoTabla("PROVINCIAS");
   assertEquals(true,respuesta > 0);
}
```

Resultado Esperado	
Para verificar que exista contenido en la tabla **PROVINCIAS**, este proceso devolverá un valor entero, el cual representa la cantidad de registros, el mismo que tiene que ser mayor a 0 si existen registros en dicha tabla	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 810ms

Realizado por: Wilmer B. 2018.

Caso de prueba 17 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_17	**Descripción:** Verificar que existan datos de la tabla “CIUDADES”
Entrada(s)	

```
@Test
public void verificarContenidoTablaCiudades() {
   int respuesta=baseDatos.verificarContenidoTabla("CIUDADES");
   assertEquals(true,respuesta > 0);
}
```

Resultado Esperado	
Para verificar que exista contenido en la tabla **CIUDADES**, este proceso devolverá un valor entero, el cual representa la cantidad de registros, el mismo que tiene que ser mayor a 0 si existen registros en dicha tabla	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 207ms

Realizado por: Wilmer B. 2018.

Caso de prueba 18 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_18	**Descripción:** Verificar que existan datos de la tabla “FAVORITO”

Entrada(s)

```
@Test
public void verificarContenidoTablaFavorito() {
   int respuesta=baseDatos.verificarContenidoTabla("FAVORITO");
   assertEquals(true,respuesta > 0);
}
```

Resultado Esperado

Para verificar que exista contenido en la tabla **FAVORITO**, este proceso devolverá un valor entero, el cual representa la cantidad de registros, el mismo que tiene que ser mayor a 0 si existen registros en dicha tabla

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 333ms

Realizado por: Wilmer B. 2018.

Caso de prueba 19 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_19	**Descripción:** Verificar eliminación de datos anteriores de la base de datos local

Entrada(s)

```
@Test
public void verificarTodasTablasAsincronizarEstenLimpias() {
   SincronizarBasesDatos sbd=new SincronizarBasesDatos(mngActivity,null);
   boolean respuesta=sbd.vaciarTablasSincronizacion(baseDatos);
   assertTrue(respuesta);
}
```

Resultado Esperado

Cuando se mande a eliminar el contenido de las tablas a sincronizar, este proceso devolverá un resultado con valor positivo **true**

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 760ms

Realizado por: Wilmer B. 2018.

Caso de prueba 20 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_20	**Descripción:** Verificar inserción de datos de la tabla “COOPERATIVA” desde el servicio web en tablas locales de la base de datos

Entrada(s)

```
@Test
public void insertarDatosTablaCOOPERATIVA() {
   ArrayList<ContentValues> lstDatos= new ArrayList<>();
   ContentValues datosCoop=new ContentValues();
   datosCoop.put(Cons.TAB_COOP_ID,1);
   datosCoop.put(Cons.TAB_COOP_NOMBRE,2);
   datosCoop.put(Cons.TAB_COOP_PAG_WEB,3);
   lstDatos.add(datosCoop);

   boolean resp=baseDatos.insertDatosTablas(lstDatos,"COOPERATIVA");
   assertTrue(resp);
}
```

Resultado Esperado

Al insertar un registro en la tabla **COOPERATIVA**, este proceso devolverá como resultado un valor positivo **true** si dicha inserción fue correcta.

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 158ms

Realizado por: Wilmer B. 2018.

Caso de prueba 21 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_21	**Descripción:** Verificar inserción de datos de la tabla “DEPENDENCIA” desde el servicio web en tablas locales de la base de datos

Entrada(s)

```
@Test
public void insertarDatosTablaDEPENDENCIA() {
   ArrayList<ContentValues> lstDatos= new ArrayList<>();
   ContentValues datosDepen=new ContentValues();
```

```
    datosDepen.put(Cons.TAB_DEP_ID,1);
    datosDepen.put(Cons.TAB_DEP_NOMBRE,"agencia baños");
    datosDepen.put(Cons.TAB_DEP_TELEFONO,"0980990010");
    datosDepen.put(Cons.TAB_DEP_CORREO,"wilkwe12@gmail.com");
    datosDepen.put(Cons.TAB_DEP_CIUDAD,"riobamba");
    datosDepen.put(Cons.TAB_DEP_PROVINCIA,"chimborazo");
    datosDepen.put(Cons.TAB_DEP_DIRECCION,"veloz y pichincha");
    datosDepen.put(Cons.TAB_DEP_DIAS_LABORABLES,1236);
    datosDepen.put(Cons.TAB_DEP_ID_COOP,1);

    boolean resp=baseDatos.insertDatosTablas(lstDatos,"DEPENDENCIA");
    assertTrue(resp);
}
```

Resultado Esperado	
Al insertar un registro en la tabla **DEPENDENCIA**, este proceso devolverá como resultado un valor positivo **true** si dicha inserción fue correcta.	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 79ms

Realizado por: Wilmer B. 2018.

Caso de prueba 22 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_22	**Descripción:** Verificar inserción de datos de la tabla "TERM_PERTENECE_DEPEN" desde el servicio web en tablas locales de la base de datos

Entrada(s)

```
@Test
public void insertarDatosTablaTERM_PERTENECE_DEPEN() {
    ArrayList<ContentValues> lstDatos= new ArrayList<>();
    ContentValues datosTPD=new ContentValues();
    datosTPD.put(Cons.TAB_TPD_ID_DEP,1);
    datosTPD.put(Cons.TAB_TPD_ID_TRM,1);
    lstDatos.add(datosTPD);
```

boolean resp=**baseDatos**.insertDatosTablas(lstDatos,**"TERM_PERTENECE_DEPEN"**); *assertTrue*(resp); }
Resultado Esperado
Al insertar un registro en la tabla **TERM_PERTENECE_DEPEN**, este proceso devolverá como resultado un valor positivo **true** si dicha inserción fue correcta.
Evaluación de la prueba: Exitosa | **Tiempo de ejecución:** 1s 229ms

Realizado por: Wilmer B. 2018.

Caso de prueba 23 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_23	**Descripción:** Verificar inserción de datos de la tabla "TERMINAL" desde el servicio web en tablas locales de la base de datos

Entrada(s)

```
@Test
public void insertarDatosTablaTERMINAL() {
   ArrayList<ContentValues> lstDatos= new ArrayList<>();
   ContentValues datosTerminal=new ContentValues();
   datosTerminal.put(Cons.TAB_TERM_ID,1);
   datosTerminal.put(Cons.TAB_TERM_NOMBRE,"terminal terrestre de riobamba");
   datosTerminal.put(Cons.TAB_TERM_DIRECCION,"av. josé antonio y daniel leon
borja");
   lstDatos.add(datosTerminal);

   boolean resp=baseDatos.insertDatosTablas(lstDatos,"TERMINAL");
   assertTrue(resp);
}
```

Resultado Esperado	
Al insertar un registro en la tabla **TERMINAL**, este proceso devolverá como resultado un valor positivo **true** si dicha inserción fue correcta.	
Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 223ms

Realizado por: Wilmer B. 2018.

Caso de prueba 24 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_24	**Descripción:** Verificar inserción de datos de la tabla "DEPEN_POSEE_RUTAS" desde el servicio web en tablas locales de la base de datos

Entrada(s)

```
@Test
public void insertarDatosTablaDEPEN_POSEE_RUTAS() {
   ArrayList<ContentValues> lstDatos= new ArrayList<>();
   ContentValues datosDPR=new ContentValues();
   datosDPR.put(Cons.TAB_DPR_ID,1);
   datosDPR.put(Cons.TAB_DPR_TURNO,1);
   datosDPR.put(Cons.TAB_DPR_COSTO,10.23);
   datosDPR.put(Cons.TAB_DPR_TIEMPO,"04:05:00");
   datosDPR.put(Cons.TAB_DPR_ID_RTA,1);
   datosDPR.put(Cons.TAB_DPR_ID_DEP,1);
   lstDatos.add(datosDPR);

   boolean resp=baseDatos.insertDatosTablas(lstDatos,"DEPEN_POSEE_RUTAS");
   assertTrue(resp);
}
```

Resultado Esperado

Al insertar un registro en la tabla **DEPEN_POSEE_RUTAS**, este proceso devolverá como resultado un valor positivo **true** si dicha inserción fue correcta.

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 234ms

Realizado por: Wilmer B. 2018.

Caso de prueba 25 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_25	**Descripción:** Verificar inserción de datos de la tabla "RUTA" desde el servicio web en tablas locales de la base de datos

Entrada(s)

```
@Test
public void insertarDatosTablaRUTA() {
   ArrayList<ContentValues> lstDatos= new ArrayList<>();
   ContentValues datosRuta=new ContentValues();
   datosRuta.put(Cons.TAB_RUTA_ID,1);
   datosRuta.put(Cons.TAB_RUTA_CIUDAD_ORIGEN,"riobamba");
   datosRuta.put(Cons.TAB_RUTA_CIUDAD_DESTINO,"quito");
   lstDatos.add(datosRuta);

   boolean resp=baseDatos.insertDatosTablas(lstDatos,"RUTA");
   assertTrue(resp);
}
```

Resultado Esperado

Al insertar un registro en la tabla **RUTA**, este proceso devolverá como resultado un valor positivo **true** si dicha inserción fue correcta.

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 934ms

Realizado por: Wilmer B. 2018.

Caso de prueba 26 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_26	**Descripción:** Verificar inserción de datos de la tabla “HORARIO” desde el servicio web en tablas locales de la base de datos

Entrada(s)

```
@Test
public void insertarDatosTablaHORARIO() {
   ArrayList<ContentValues> lstDatos= new ArrayList<>();
   ContentValues datosHorario=new ContentValues();
   datosHorario.put(Cons.TAB_HORARIO_ID,1);
   datosHorario.put(Cons.TAB_HORARIO_HORA_SALIDA,"12:05:00");
   datosHorario.put(Cons.TAB_HORARIO_ID_DPR,1);
   lstDatos.add(datosHorario);

   boolean resp=baseDatos.insertDatosTablas(lstDatos,"HORARIO");
```

assertTrue(resp); }
Resultado Esperado
Al insertar un registro en la tabla **HORARIO**, este proceso devolverá como resultado un valor positivo **true** si dicha inserción fue correcta
Evaluación de la prueba: Exitosa | **Tiempo de ejecución:** 1s 508ms

Realizado por: Wilmer B. 2018.

Caso de prueba 27 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_27	**Descripción:** Verificar inserción de datos de la tabla "CIUDAD" desde el servicio web en tablas locales de la base de datos

Entrada(s)

```
@Test
public void insertarDatosTablaCIUDADES() {
   ArrayList<ContentValues> lstDatos= new ArrayList<>();
   ContentValues datosCiudades=new ContentValues();
   datosCiudades.put(Cons.TAB_CIUDAD_ID,1);
   datosCiudades.put(Cons.TAB_CIUDAD_NOMBRE,"guayaquil");
   datosCiudades.put(Cons.TAB_CIUDAD_LATITUD,-2.1774458);
   datosCiudades.put(Cons.TAB_CIUDAD_LONGITUD,-79.9591627);
   datosCiudades.put(Cons.TAB_CIUDAD_ID_PRV,1);
   lstDatos.add(datosCiudades);


   boolean resp=baseDatos.insertDatosTablas(lstDatos,"CIUDADES");
   assertTrue(resp);
}
```

Resultado Esperado

Al insertar un registro en la tabla **CIUDADES**, este proceso devolverá como resultado un valor positivo **true** si dicha inserción fue correcta.

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 583ms

Realizado por: Wilmer B. 2018.

Caso de prueba 28 de historia de usuario 17.

CASO DE PRUEBA AUTOMATIZADA	
Sprint No: 09	**Funcionalidad:** HU_17 Sincronización de app móvil con base de datos alojado en servicio de hosting
Código: TC_28	**Descripción:** Verificar inserción de datos de la tabla “PROVINCIA” desde el servicio web en tablas locales de la base de datos

Entrada(s)

```
@Test
public void insertarDatosTablaPROVINCIAS() {
   ArrayList<ContentValues> lstDatos= new ArrayList<>();
   ContentValues datosProvincias=new ContentValues();
   datosProvincias.put(Cons.TAB_PROVINCIA_ID,1);
   datosProvincias.put(Cons.TAB_PROVINCIA_NOMBRE,"chimborazo");
   lstDatos.add(datosProvincias);

   boolean resp=baseDatos.insertDatosTablas(lstDatos,"PROVINCIAS");
   assertTrue(resp);
}
```

Resultado Esperado

Al insertar un registro en la tabla **PROVINCIAS**, este proceso devolverá como resultado un valor positivo **true** si dicha inserción fue correcta.

Evaluación de la prueba: Exitosa	**Tiempo de ejecución:** 1s 409ms

Realizado por: Wilmer B. 2018.

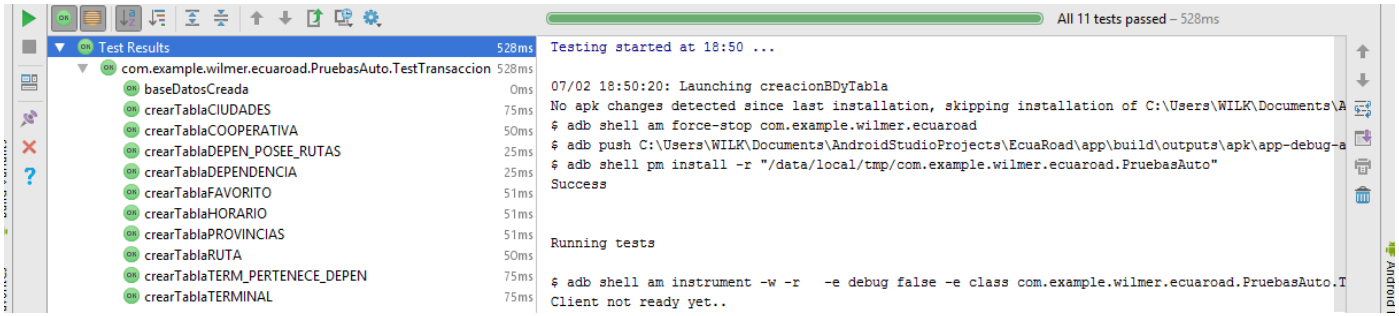
Tabla de resultados en pruebas automáticas del sprint 9 historia de usuario 17.

RESULTADO	
Cantidad de pruebas	28
Tiempo total en ejecución de pruebas Sprint 9, HU_17	41s 689ms

Realizado por: Wilmer B. 2018.

Anexo D: Caminos de evaluaciones gestionadas por la suite de Junit

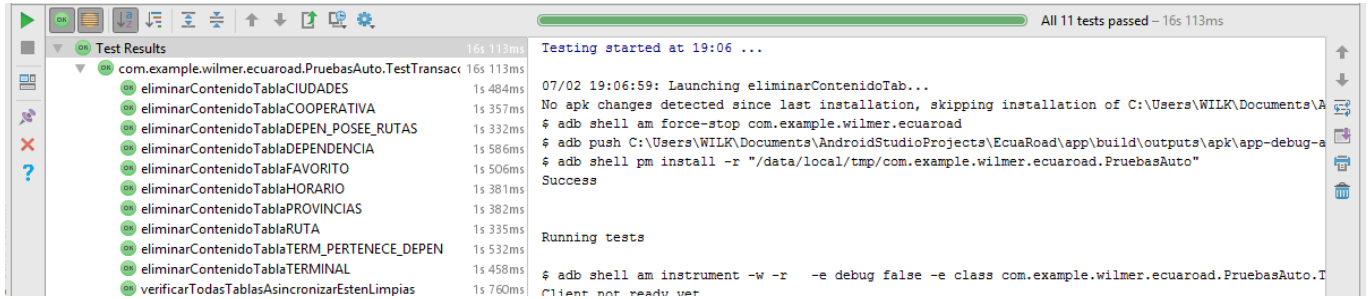
Camino de evaluación: Creación de base de datos y sus tablas



Camino de evaluación - Creación de base de datos y sus tablas.

Realizado por: Wilmer B. 2018.

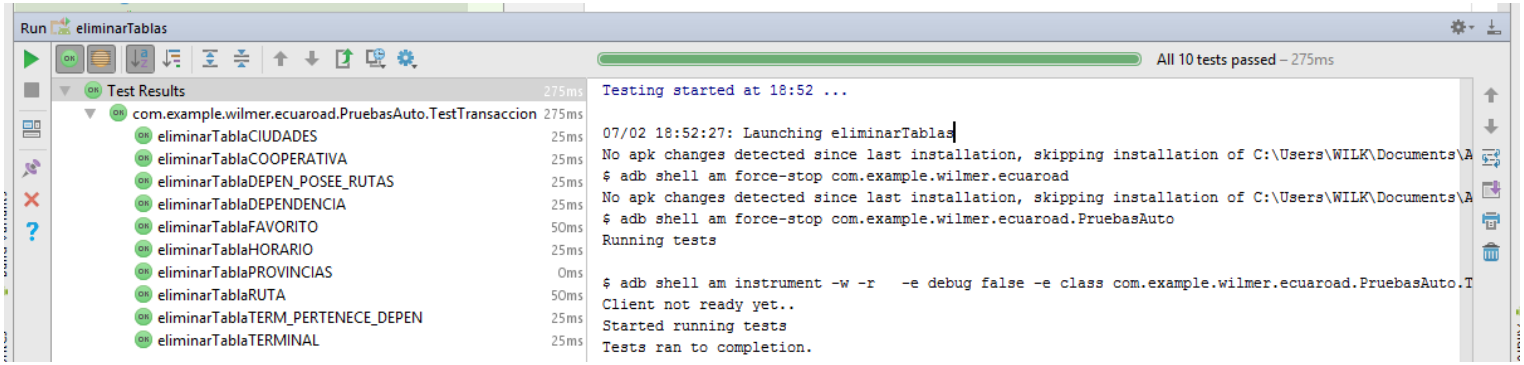
Camino de evaluación: Eliminar contenido de tablas



Camino de evaluación - Eliminar contenido de tablas.

Realizado por: Wilmer B. 2018.

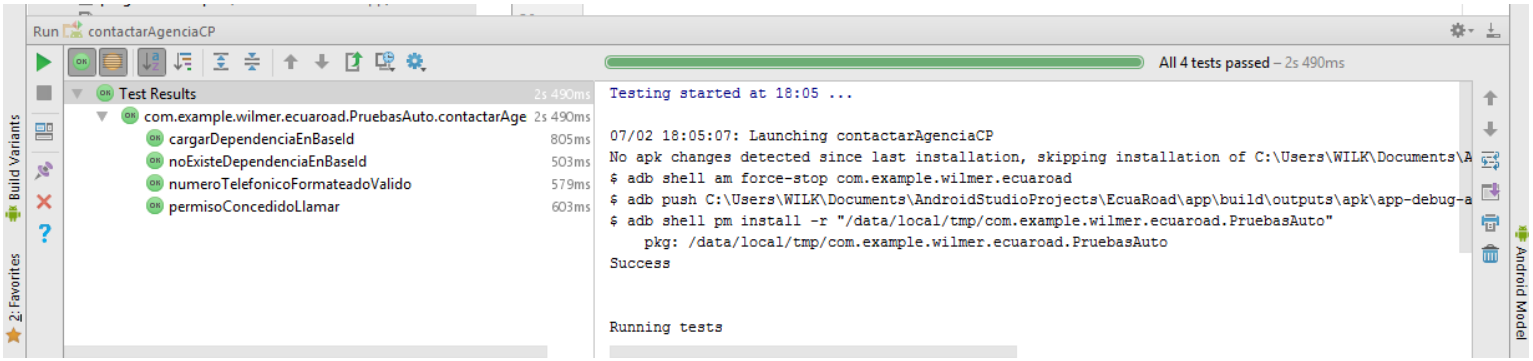
Camino de evaluación: Eliminación de tablas de la base de datos



Camino de evaluación - Eliminación de tablas de la base de datos.

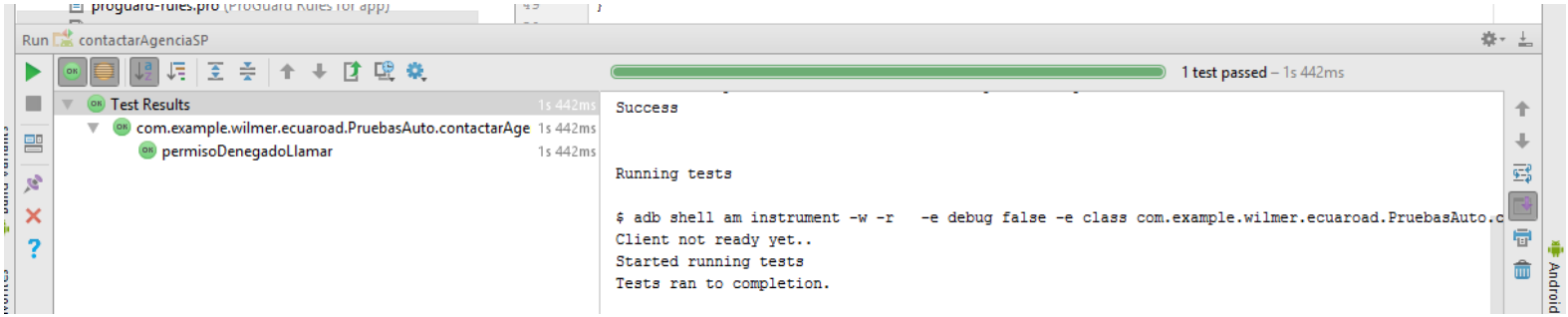
Realizado por: Wilmer B. 2018.

Camino de evaluación: Contactar Agencias de Transporte



Camino de evaluación - Contactar Agencias de Transporte (Con permisos).

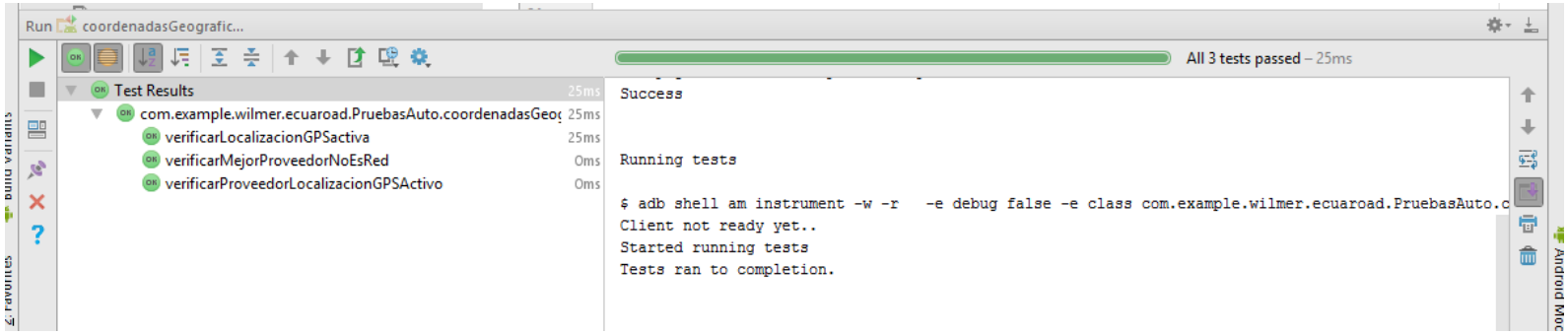
Realizado por: Wilmer B. 2018.



Camino de evaluación - Contactar Agencias de Transporte (Sin permisos).

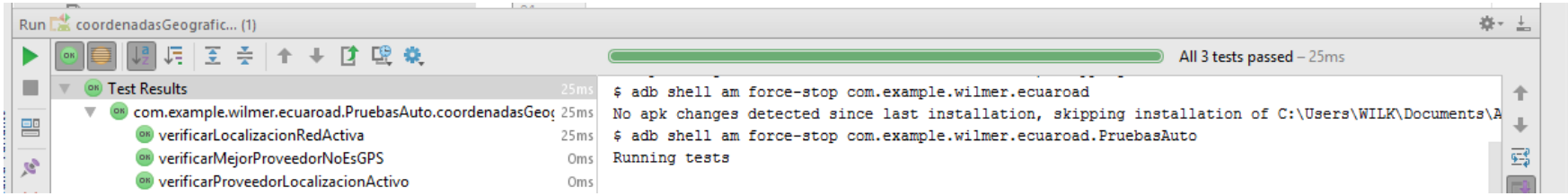
Realizado por: Wilmer B. 2018.

Camino de evaluación: Obtener coordenadas geográficas



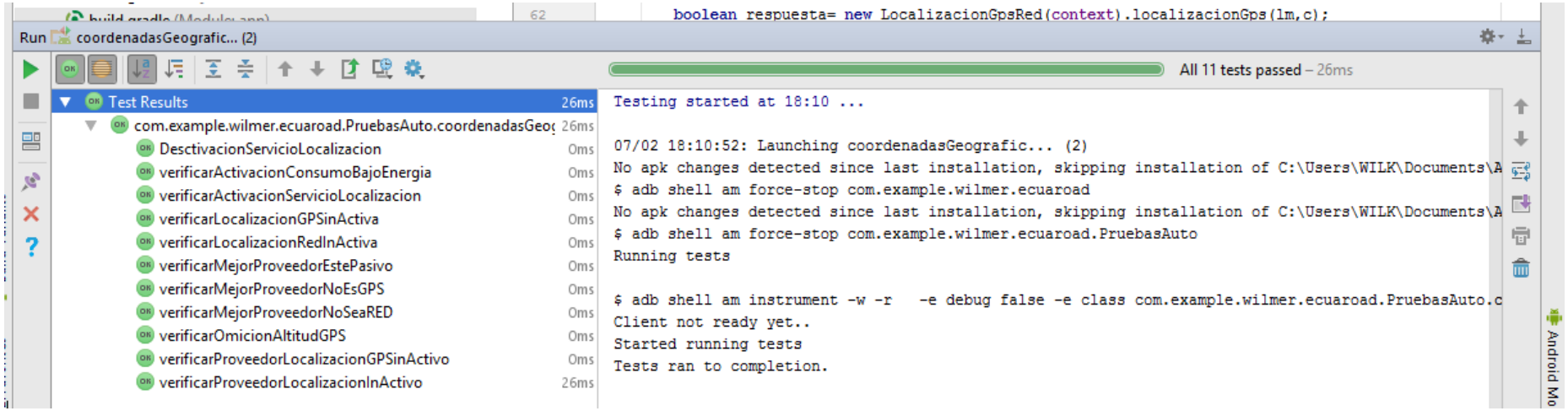
Camino de evaluación - Obtener coordenadas geográficas con GPS.

Realizado por: Wilmer B. 2018.



Camino de evaluación - Obtener coordenadas geográficas a través de red.

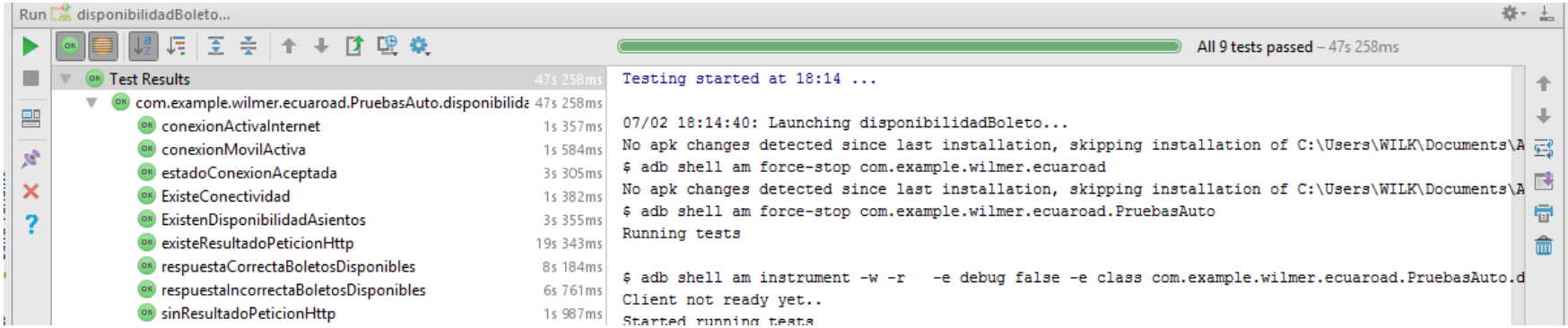
Realizado por: Wilmer B. 2018.



Camino de evaluación - Obtener coordenadas geográficas (No existe servicios de localización activos).

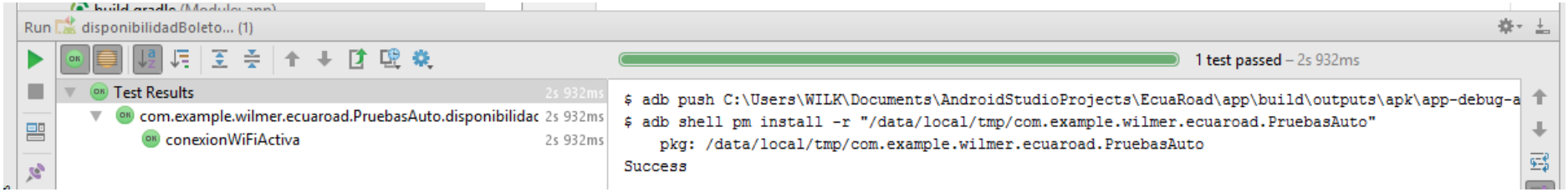
Realizado por: Wilmer B. 2018.

Camino de evaluación: Obtener disponibilidad de boletos en cooperativas de transporte



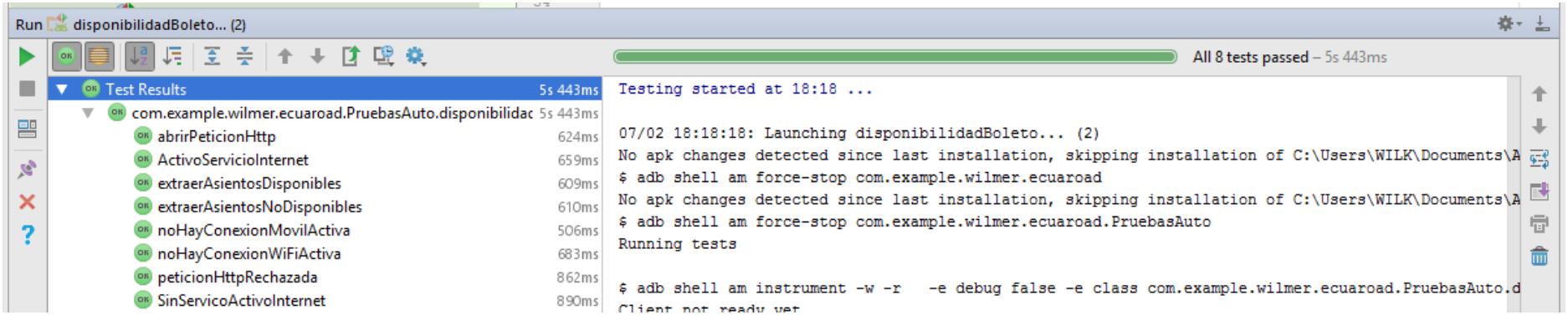
Camino de evaluación - Obtener disponibilidad de boletos en cooperativas de transporte.

Realizado por: Wilmer B. 2018.



Camino de evaluación - Obtener disponibilidad de boletos en cooperativas de transporte (Verificación de Wi-Fi).

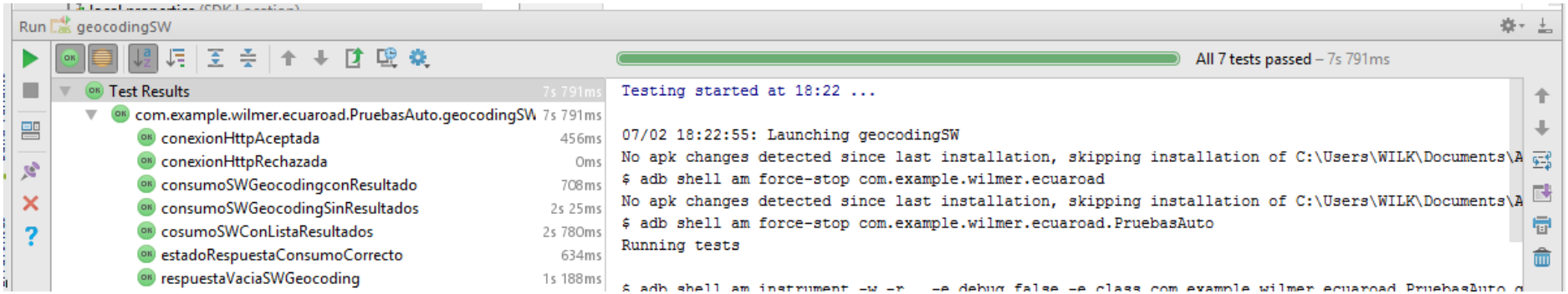
Realizado por: Wilmer B. 2018.



Camino de evaluación - Obtener disponibilidad de boletos en cooperativas de transporte (Proceso).

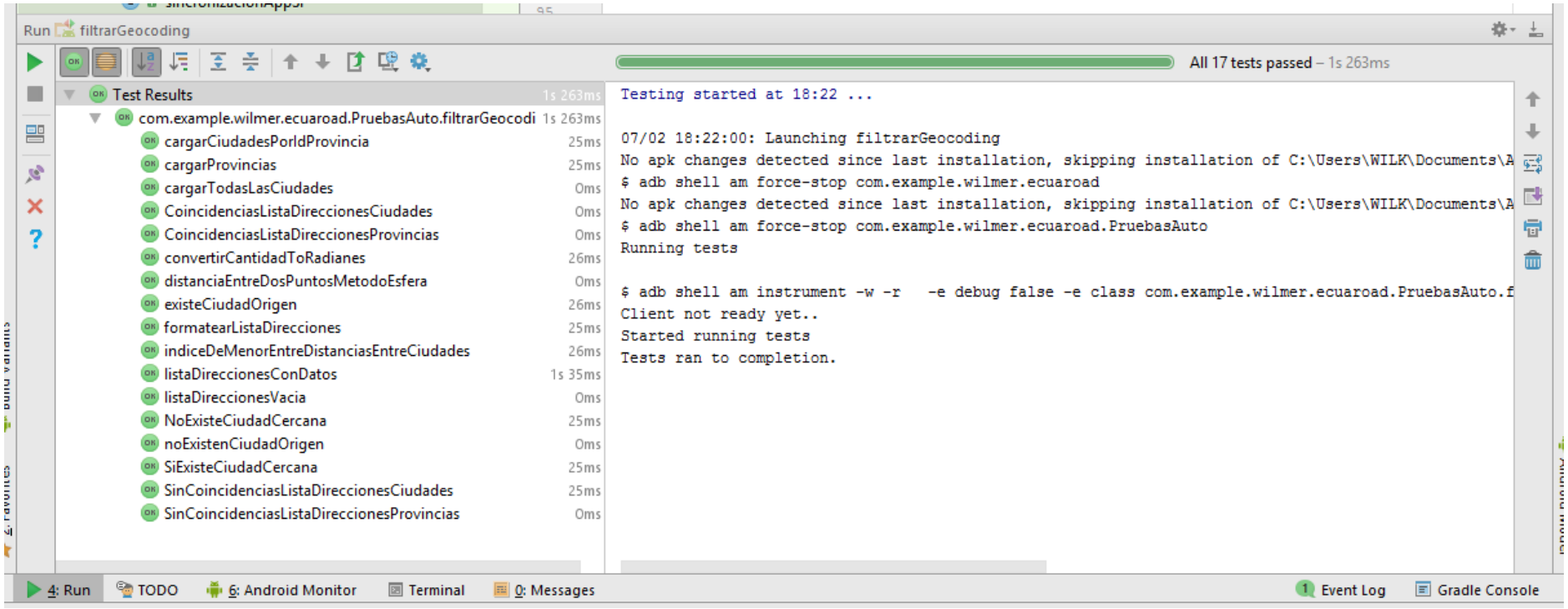
Realizado por: Wilmer B. 2018.

Camino de evaluación: Filtrar respuesta de servicio web geocoding



Camino de evaluación: Filtrar respuesta de servicio web geocoding.

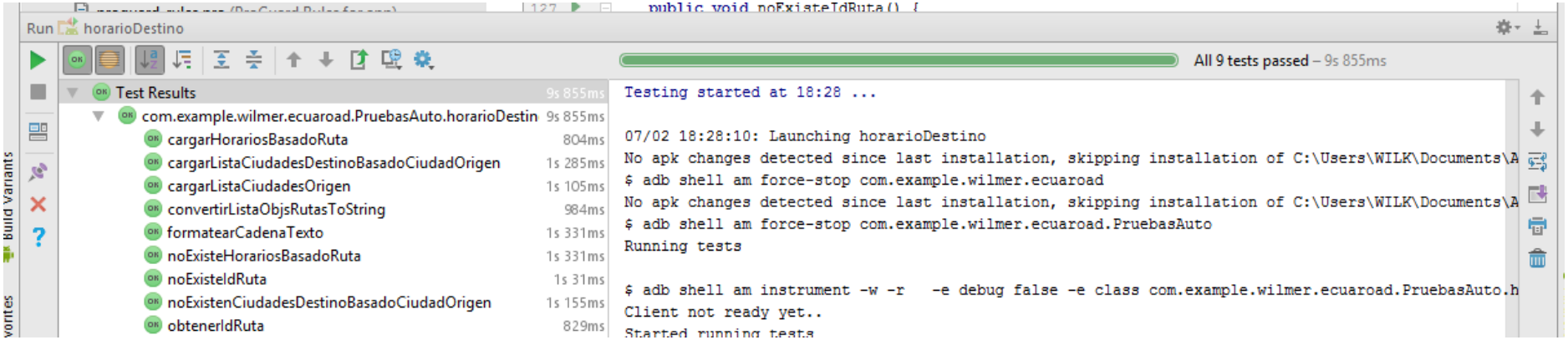
Realizado por: Wilmer B. 2018.



Camino de evaluación - Filtrar respuesta de servicio web geocoding (Procesamiento de datos).

Realizado por: Wilmer B. 2018.

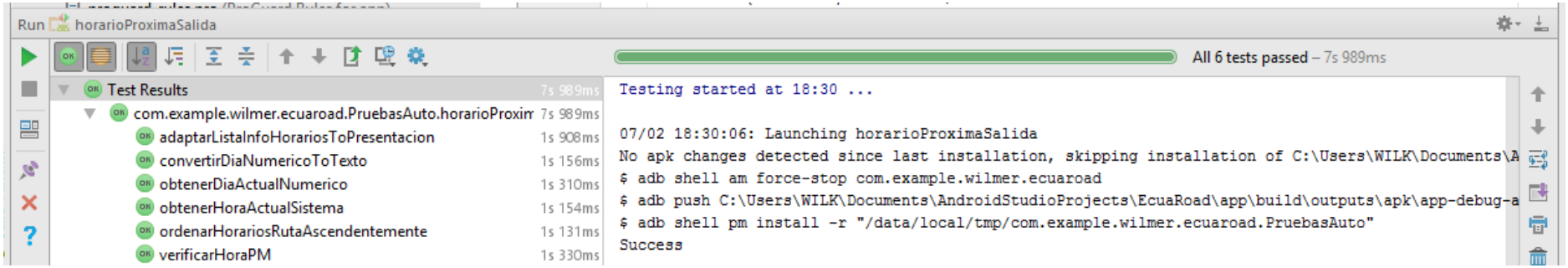
Camino de evaluación: Mostrar horarios por destino



Camino de evaluación - Mostrar horarios por destino.

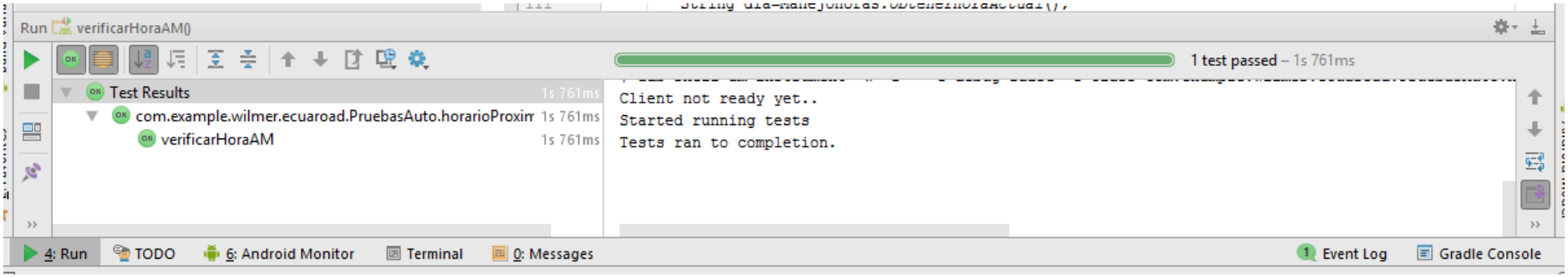
Realizado por: Wilmer B. 2018.

Camino de evaluación: Mostrar horario de próxima salida



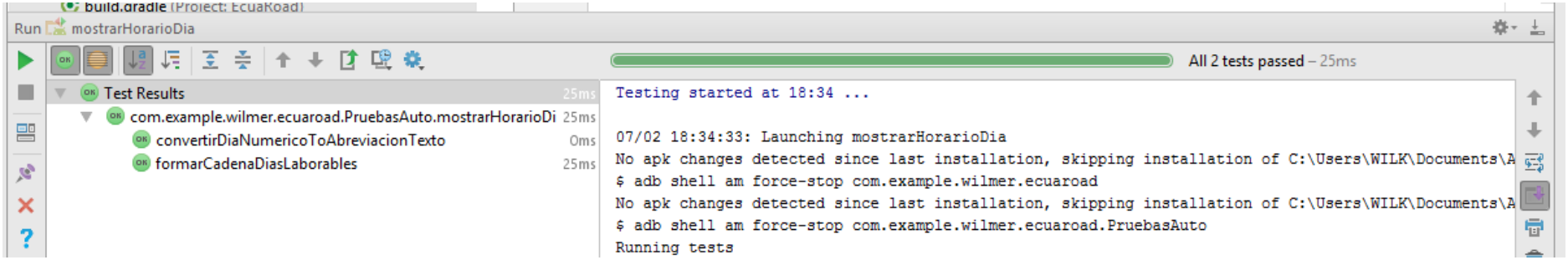
Camino de evaluación - Mostrar horario de próxima salida (Procesamiento de hora).

Realizado por: Wilmer B. 2018.



Camino de evaluación - Mostrar horario de próxima salida (Verificación de horario matutino).

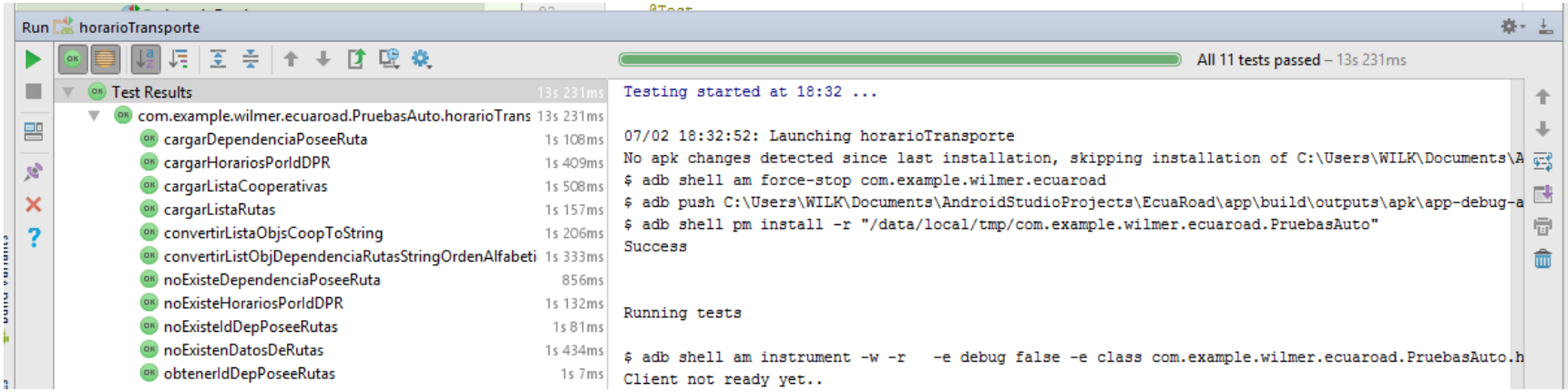
Realizado por: Wilmer B. 2018.



Camino de evaluación - Mostrar horario de próxima salida (Formatear textos).

Realizado por: Wilmer B. 2018.

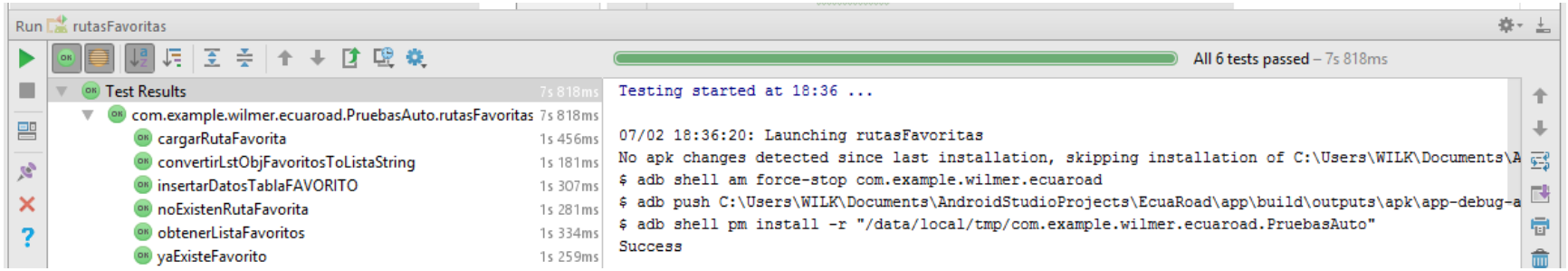
Camino de evaluación: Mostrar horarios por cooperativas de transporte



Camino de evaluación - Mostrar horarios por cooperativas de transporte.

Realizado por: Wilmer B. 2018.

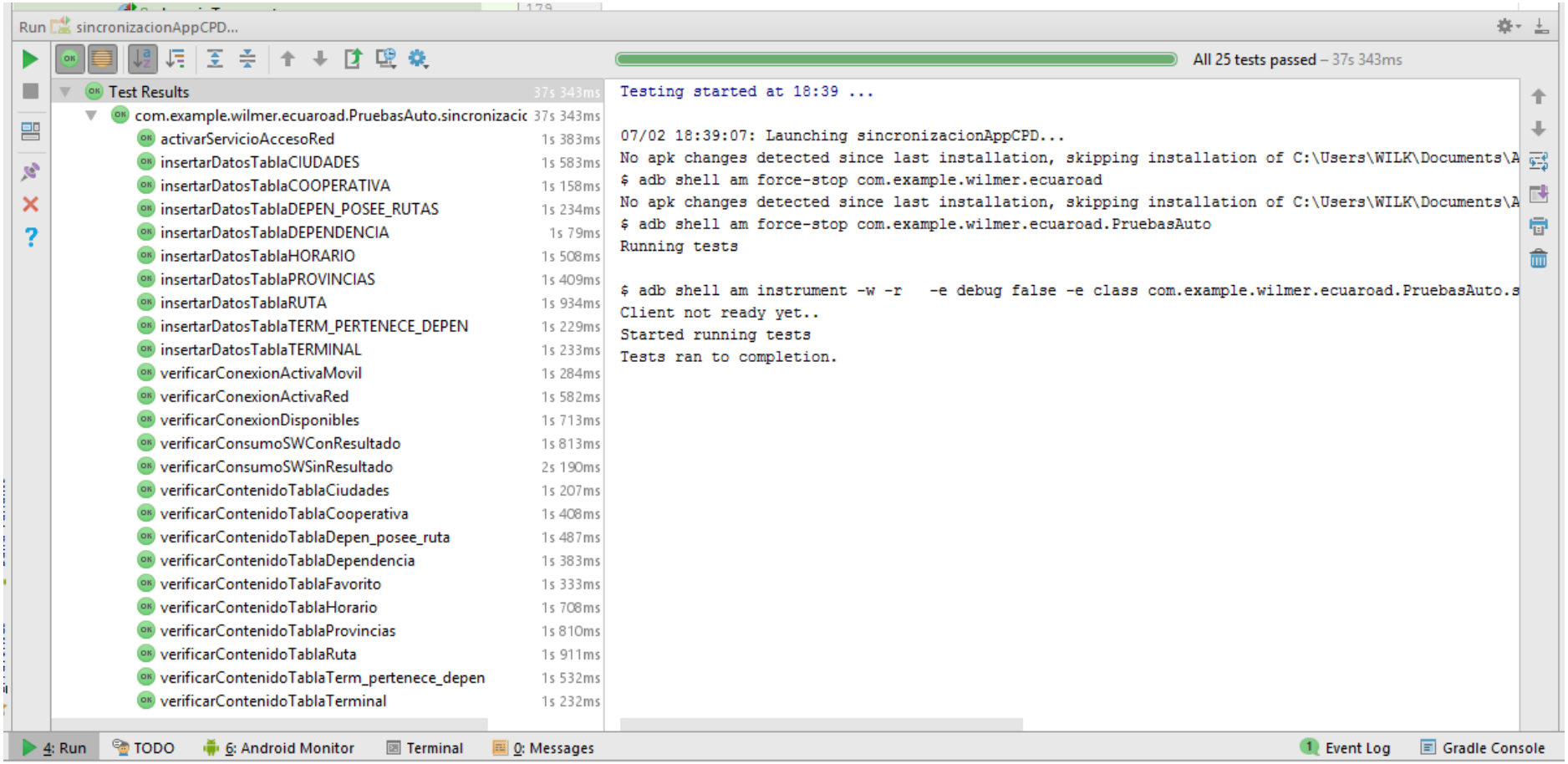
Camino de evaluación: Mostrar rutas favoritas



Camino de evaluación - Mostrar rutas favoritas.

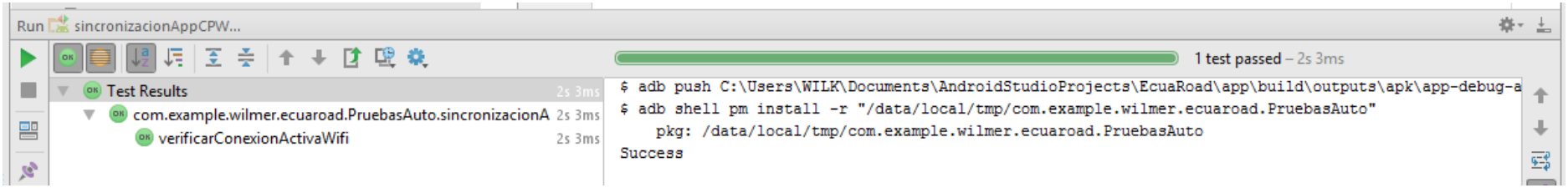
Realizado por: Wilmer B. 2018.

Camino de evaluación: Sincronización de base de datos del hosting con base de datos local.



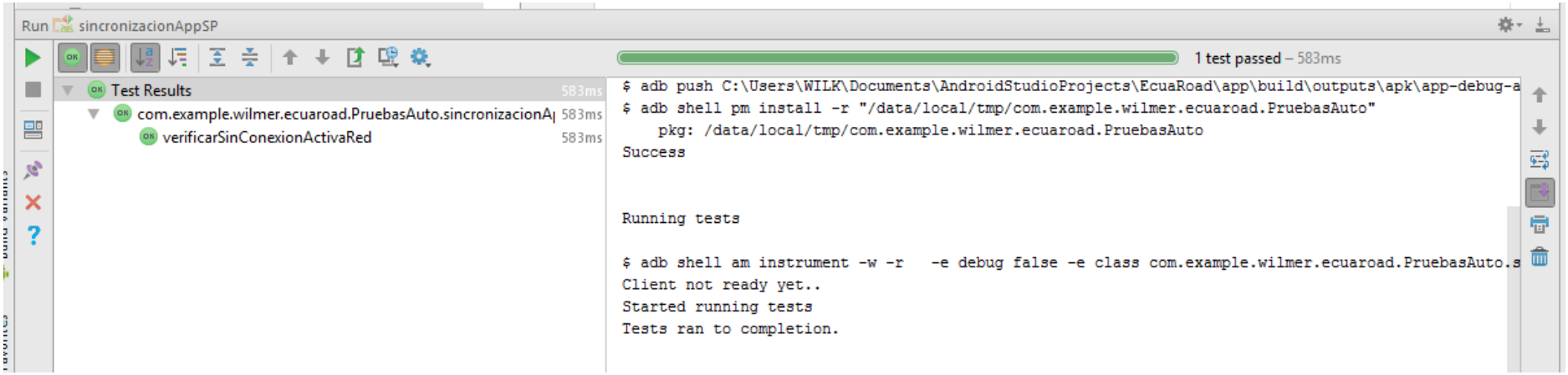
Camino de evaluación - Sincronización de base de datos del hosting con base de datos local (Inserción de datos).

Realizado por: Wilmer B. 2018.



Camino de evaluación - Sincronización de base de datos del hosting con base de datos local (Verificación de conexión activa).

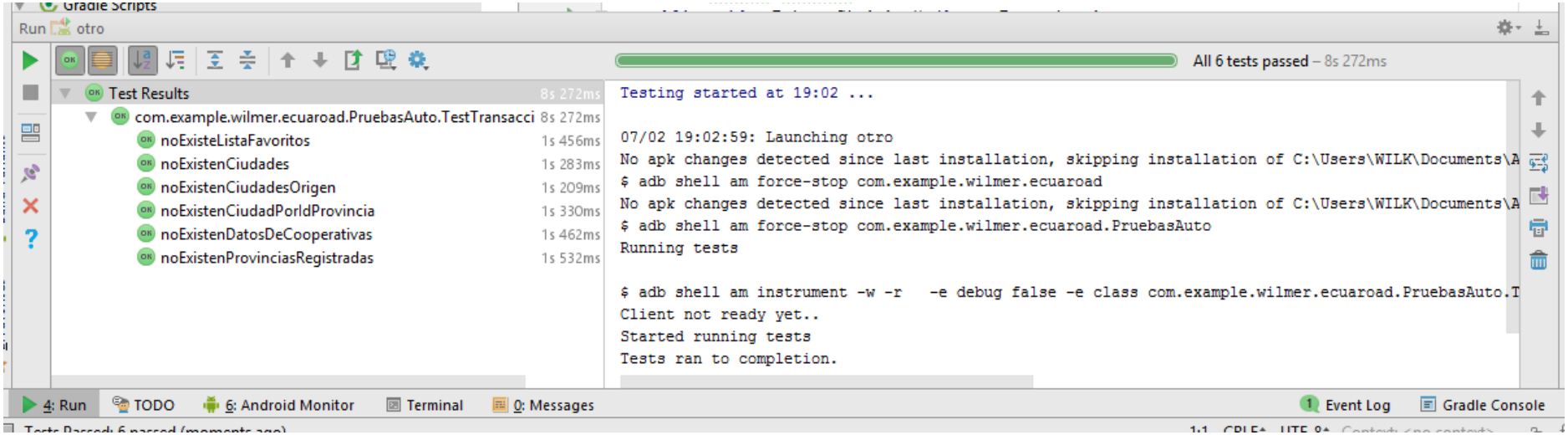
Realizado por: Wilmer B. 2018.



Camino de evaluación - Sincronización de base de datos del hosting con base de datos local (Verificación conexión a red activa).

Realizado por: Wilmer B. 2018.

Camino de evaluación: Consultas sin resultados



Camino de evaluación - Consultas sin resultados.

Realizado por: Wilmer B. 2018.

Anexo E: Manual de Usuario

APLICAIÓN MÓVIL NIS

El presente documento tiene como finalidad de mostrar la funcionalidad de la aplicación móvil NIS.

1. Pantalla de instalación

Es la primera pantalla que se muestra cuando se abre la aplicación, una vez esta haya sido instalada, esta consta de dos botones SALIR y SINCRONIZAR. Si presionamos el botón SALIR nos llevará fuera de la aplicación móvil.

Pantalla de instalación.

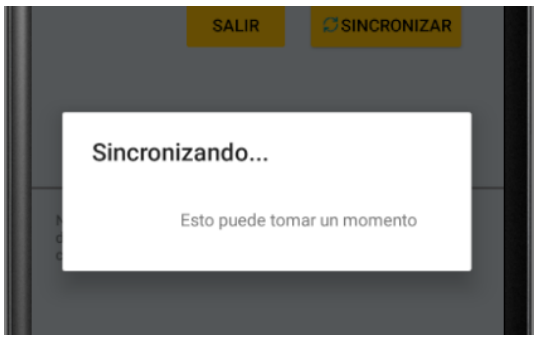
Realizado por: Wilmer B. 2018.

En el caso de presionar el botón SINCRONIZAR el sistema verificará que esta sincronización se lleva automáticamente en caso de estar conectado al internet por medio de una conexión a una red Wi-Fi, en caso de no ser así, se mostrará un cuadro de diálogo en el que se da a conocer las opciones de conexión para llevar a cabo la sincronización de la información, si se presiona en el botón ATIVAR WI FI se activará esta conexión, caso contrario si se presiona el botón SEGUIR CON DATOS, cuando se elige esta opción se realiza la sincronización con datos móviles, lo que puede generar costos adicionales con la operadora.

Advertencia - Conexión a la red con datos móviles.

Realizado por: Wilmer B. 2018.

Con cualquiera de las opciones de sincronización, el sistema procederá a realizar la sincronización de la informaión, esto se dará a conocer mostrando un cuadro de diálogo donde nos dice que espermos un momento mientras pasa este proceso de sincronización.



Procesando sincronización de pantalla de instalación.

Realizado por: Wilmer B. 2018.

Una vez que se haya sincronizado la información, se procederá a mostrar la pantalla de inicio junto con un mensaje advirtiendo que la sincronización de datos fue exitosa.

Mensaje de sincronización exitosa.

Realizado por: Wilmer B. 2018.

2. Pantalla de inicio

A continuación se muestra la primera pantalla de la aplicación, la cual será presentada cada vez que se abra la aplicación en un dispositivo, la cual consta de: una imagen de logo de la aplicación, además de 4 botones, los botones ACERCA DE y MÁS son botones que contienen información estática de la empresa sin nigún tipo de funcionalidad, mientras que el botón BUSCAR es el botón que contiene la mayoría de funcionalidades que fueron desarrolladas en este proyecto y es donde se encuentra el menú de búsqueda de horarios. Otro de los botones es el de FAVORITOS en donde se encuentra la lista de rutas favoritas con su respectivo horario.

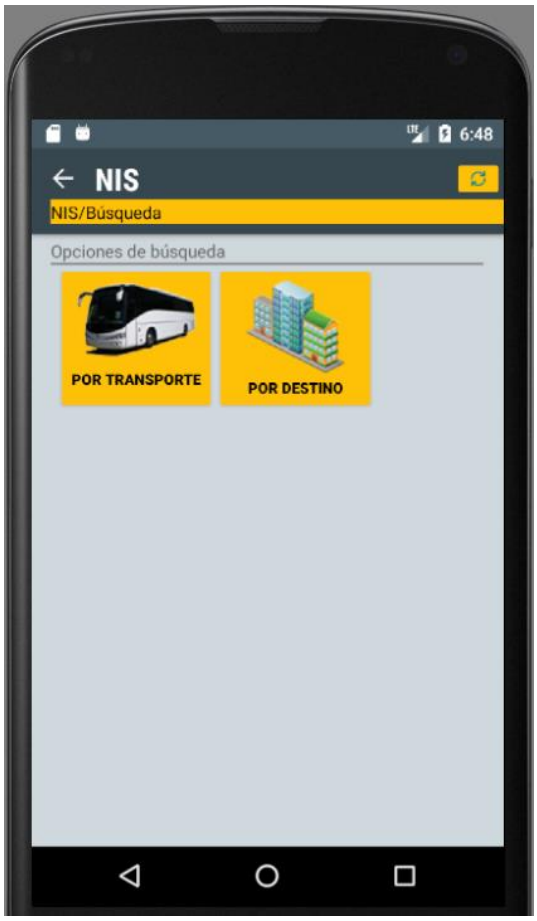
Pantalla de inicio.

Realizado por: Wilmer B. 2018.

3. Menú de búsquedas

Dentro de este menú se encuentran lo siguientes elementos: 

, esta es una flecha de retroceso jerarquico en las pantallas que se han ido abriendo dentro de la aplicación,  este botón se ubicará en cada una de las pantallas de la aplicaión, excepto la de inicio, este servirá para sincronizar la información de horarios con el fin de obtener a información más precisa y actual. También se encuentran dos opciones de menú, la búsqueda por cooperativa de transporte y la búsqueda por ruta basada en una ciudad origen y destino.

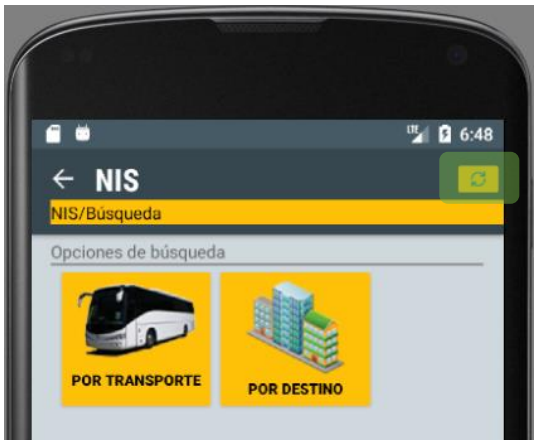


Menú de búsquedas.

Realizado por: Wilmer B. 2018.

4. Sincronización de bases de datos

Al presionar el botón  , se comenzará con el proceso que conlleva la sincronización, para esto se deberá estar conectado a internet en caso de que no sea así el sistema verificará que esto se lleve a cabo antes de sincronizar la información.



Sincronización desde pantalla de menú de búsquedas.

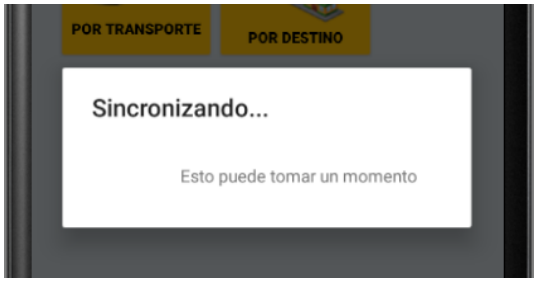
Realizado por: Wilmer B. 2018.

Una vez que presionamos el botón sincronizar, el sistema verificará el tipo de conectividad que se tiene hacia internet, en caso de estar conectado a una red wifi la sincronización seguirá con su proceso, pero en caso de estar conectado por medio de datos móviles entonces el sistema mostrará un cuadro de alerta indicando sobre este tipo de conexión que puede incurrir en gastos con la operadora con la que trabaja, dando dos opciones para poder seguir con el proceso de sincronización ACTIVAR WI FI y SEGUIR CON DATOS, al presionar el botón activar wifi se activará la conexión WI-FI, mientras que si se presiona el botón SEGUIR CON DATOS, se procederá a sincronizar consumiendo datos móviles.

Advertencia consumo de datos móviles.

Realizado por: Wilmer B. 2018.

Una vez elegido el tipo de conexión por la cual se sincronizará mostrará un cuadro de espera, advirtiendo al usuario que debe esperar mientras se sincroniza la información.



Procesamiento de sincronización.

Realizado por: Wilmer B. 2018.

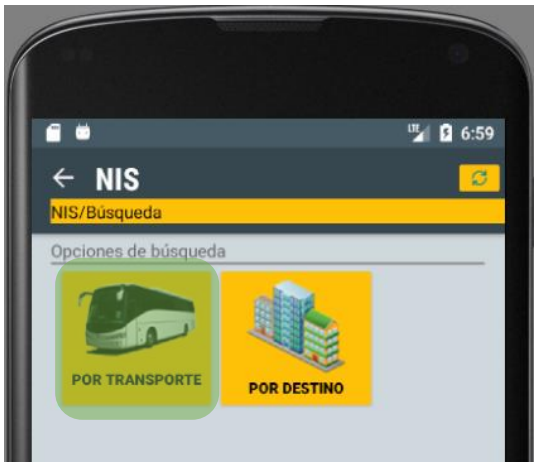
Una vez que la información sea sincronizada mostrará un mensaje en pantalla advirtiendo que la sincronización fue exitosa.

Sincronización exitosa

Realizado por: Wilmer B. 2018.

5. Búsqueda por cooperativa de transporte

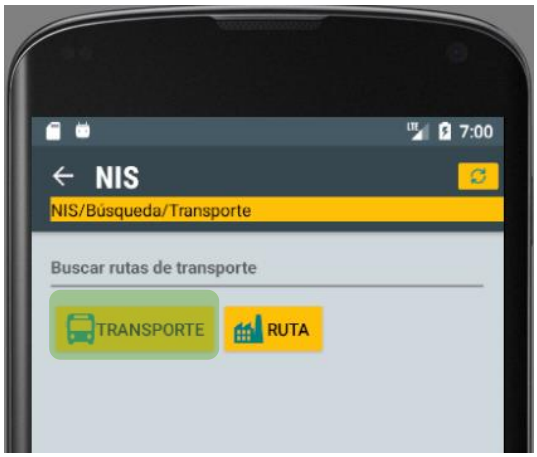
Para elegir la búsqueda por transporte debemos dar clic en la opción POR TRANSPORTE, y nos mostrará la interfaz de búsqueda.



Búsqueda de horarios por cooperativa de transporte.

Realizado por: Wilmer B. 2018.

A continuación, se muestra la pantalla de búsqueda por transporte, esta contiene dos botones TRANSPORTE y RUTA, el botón TRANSPORTE sirve para buscar todas las cooperativas de transporte disponibles, mientras que el botón RUTA muestra la lista de rutas cubiertas por una cooperativa de transporte específica.



Listar cooperativas de transporte.

Realizado por: Wilmer B. 2018.

Cuando presionamos el botón TRANSPORTE, se presentará en la pantalla un cuadro de diálogo con una la lista de cooperativas de transporte ordenadas alfabéticamente.

Lista de cooperativas de transporte ordenada.

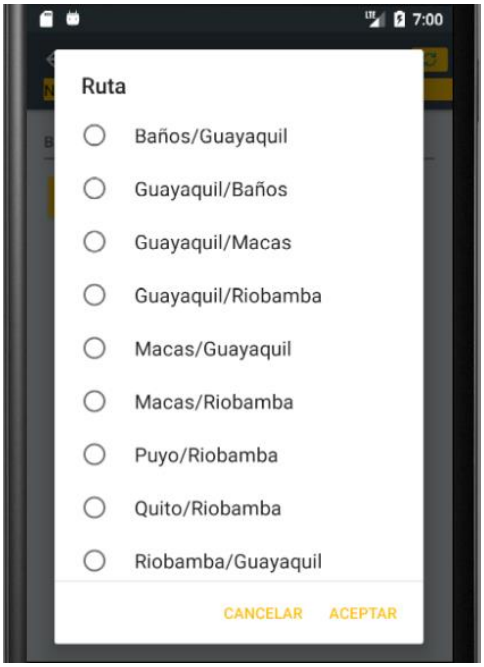
Realizado por: Wilmer B. 2018.

Dentro de esta lista debemos seleccionar una cooperativa de transporte y presionar la opción ACEPTAR.

Selección de cooperativa de transporte.

Realizado por: Wilmer B. 2018.

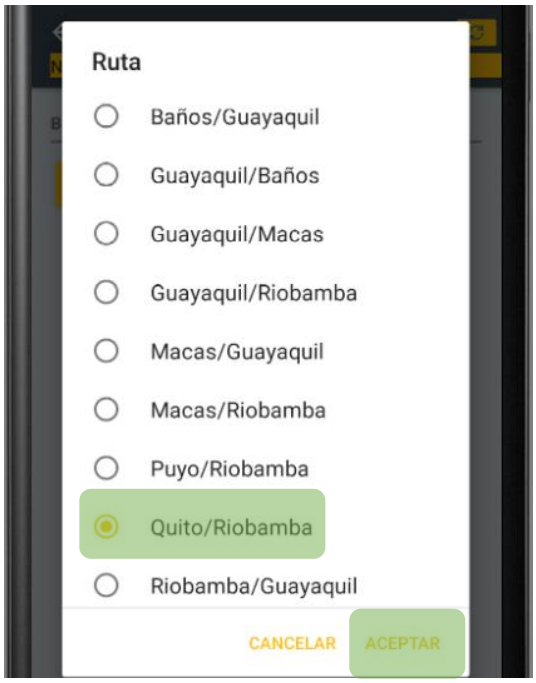
Ahora nos mostrará la lista de rutas que cubre dicha cooperativa de transporte igualmente ordenadas alfabéticamente.



Lista de rutas cubiertas por cooperativa de transporte.

Realizado por: Wilmer B. 2018.

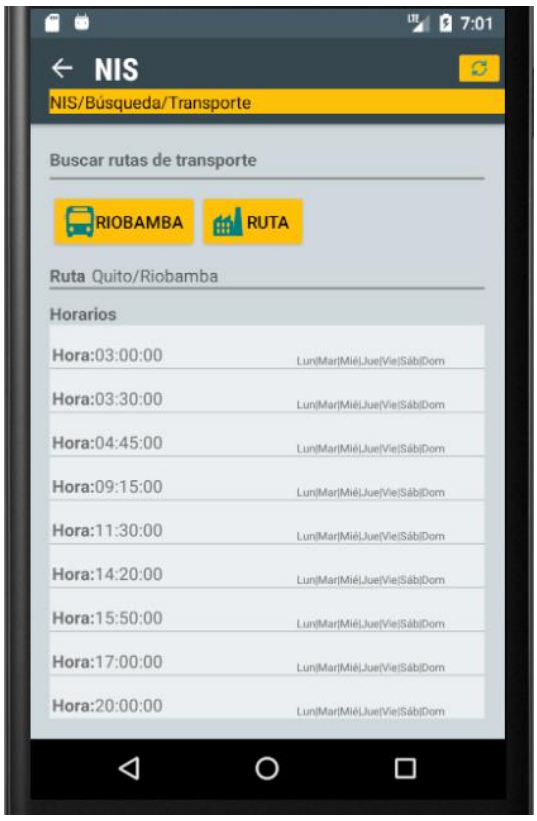
Para poder elegir una ruta debemos desplazarnos dentro de este cuadro de dialogo y encontrar la ruta deseada, la seleccionaremos y presionaremos el botón ACEPTAR.



Selección de ruta específica.

Realizado por: Wilmer B. 2018.

Una vez elegida la ruta, se mostrará la lista de horarios disponibles que cubren dicha ruta ordenados por hora.



Lista de horarios buscados por cooperativa de transporte.

Realizado por: Wilmer B. 2018.

6. Búsqueda de horarios con base en una ciudad origen y destino

Para elegir la búsqueda basada en una ciudad origen y destino se debe elegir la opción POR DESTINO.

Búsqueda por destino

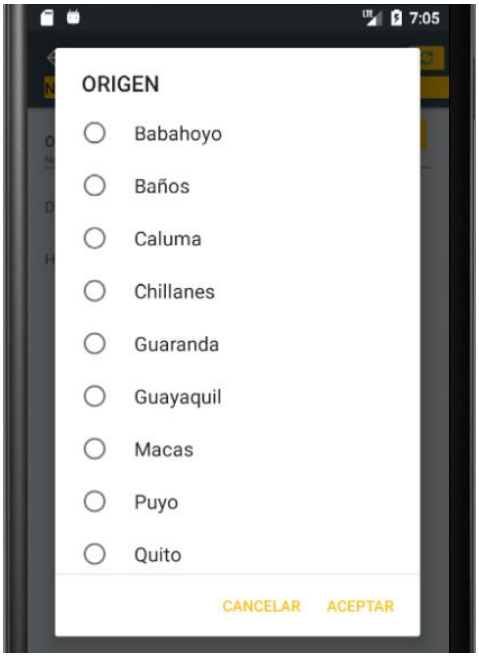
Realizado por: Wilmer B. 2018.

Esta será la pantalla que se muestre al elegir la opción de búsqueda POR DESTINO, en la cual consta de 3 botones, dos botones para elegir una ciudad, el primero elige la ciudad origen desde la cual se verificará el inicio de la ruta, mientras que el segundo botón busca la ciudad de destino de la ruta, y por último el botón de  (geolocalización). Este botón busca automáticamente la ciudad en la que se encuentra este dispositivo.

Listar ciudades origen.

Realizado por: Wilmer B. 2018.

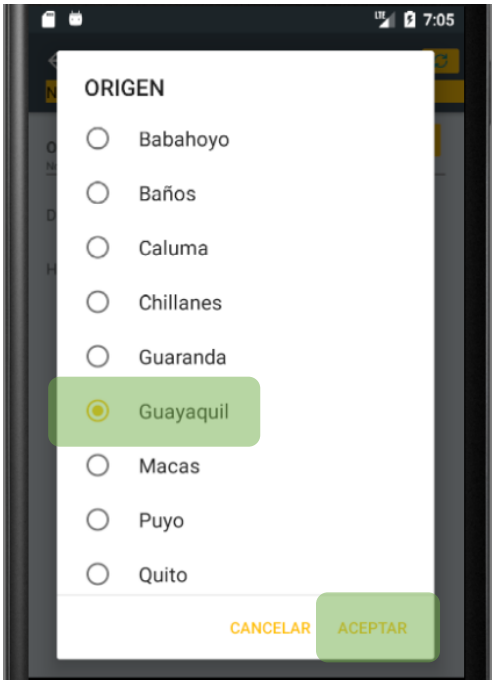
Al presionar el botón de búsqueda de ciudad origen o DESDE, se presenta la lista de ciudades origen de una ruta, estas están ordenadas alfabéticamente.



Lista de ciudades origen ordenadas.

Realizado por: Wilmer B. 2018.

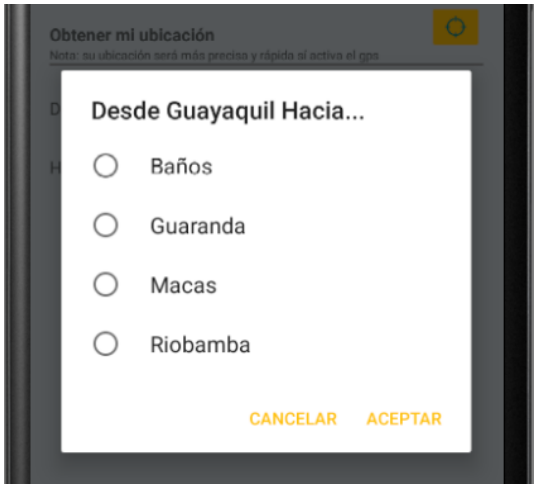
Dentro de esta lista debemos elegir la ciudad deseada y luego presionar el botón ACEPTAR.



Selección de ciudad origen.

Realizado por: Wilmer B. 2018.

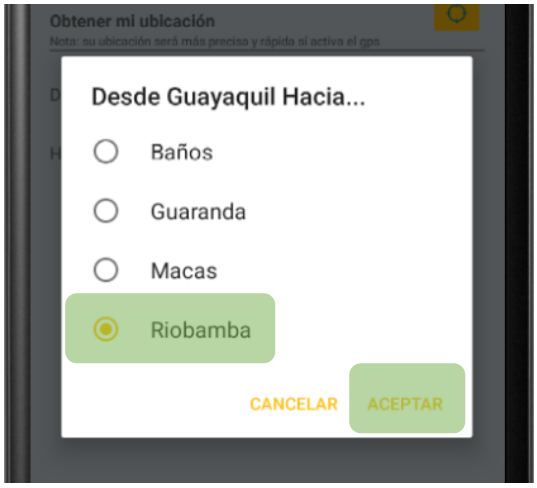
Seguido de esto se presenta automáticamente la lista de ciudades destino que posee la ciudad origen antes seleccionada. La cual de la misma manera se encuentra ordenada alfabéticamente.



Lista de ciudades destino ordenadas.

Realizado por: Wilmer B. 2018.

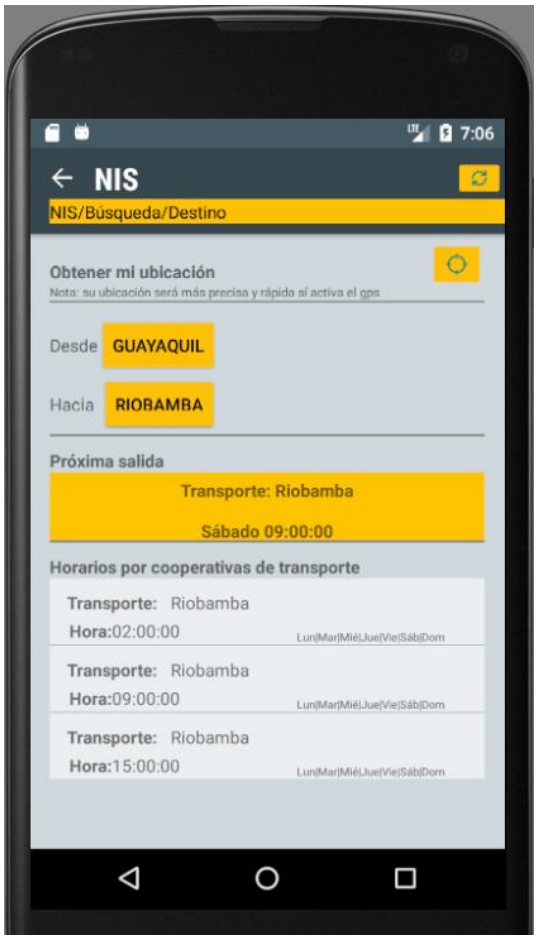
De esta lista elegiremos la ciudad destino y presionaremos la opción ACEPTAR.



Selección de ciudad destino.

Realizado por: Wilmer B. 2018.

Una vez se haya elegido la ruta (ciudad origen y destino) se procederá a mostrar la lista de horarios ordenados, también se mostrará la salida más próxima con respecto a la hora actual del dispositivo.



Lista de horarios de una ruta por búsqueda por destino.

Realizado por: Wilmer B. 2018.

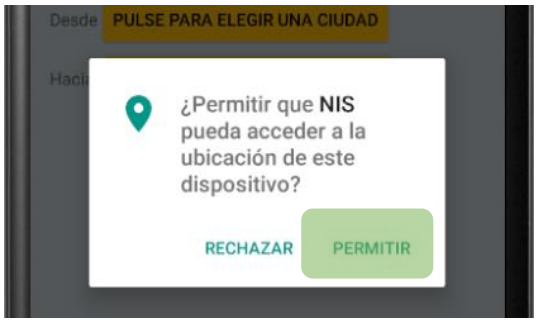
7. Búsqueda de horarios con base en ciudad origen y destino a través de geo loalización.

Para usar la opción de geolocalización debemos entrar a la opción de búsqueda por DESTINO, en esta opción debemos presionar el botón sincronizar  .

Localización de ciudad donde se encuentra el dispositivo.

Realizado por: Wilmer B. 2018.

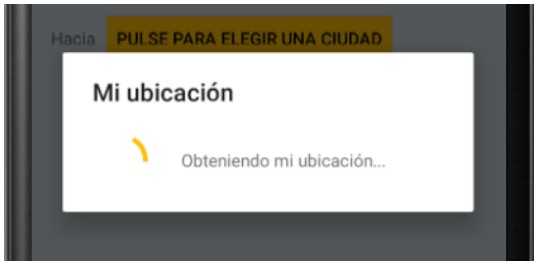
A continuación, una pantalla nos pedirá permisos para poder llevará cabo el proceso de localización en el cual podemos RECHAZAR o PERMITIR. Cuando se elige la opción RECHAZAR no se concederán permisos para usar esta funcionalidad, mientras que al presionar PERMITIR se procederá con el proceso de localización.



Petición de permisos de ubicación.

Realizado por: Wilmer B. 2018.

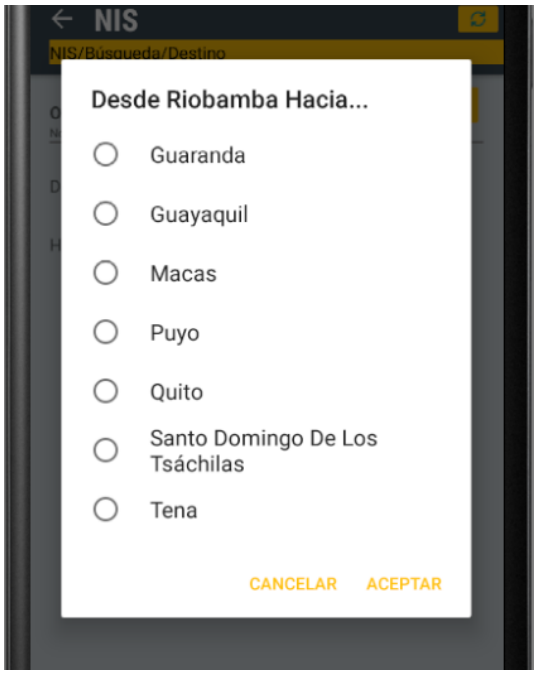
Una vez el permiso haya sido concedido se mostrará un cuadro de proceso que indica que la localización se está llevado a cabo.



Procesamiento de ubicación.

Realizado por: Wilmer B. 2018.

Una vez que se haya detectado la ciudad en la que se encuentra, mostrará la lista de ciudades destino que posee dicha ciudad origen encontrada por medio de geolocalización, de la cual debemos elegir y seguir el proceso anterior para buscar los horarios de una ciudad destino.



Lista de ciudades destino en base a ciudad origen geolocalizada.

Realizado por: Wilmer B. 2018.

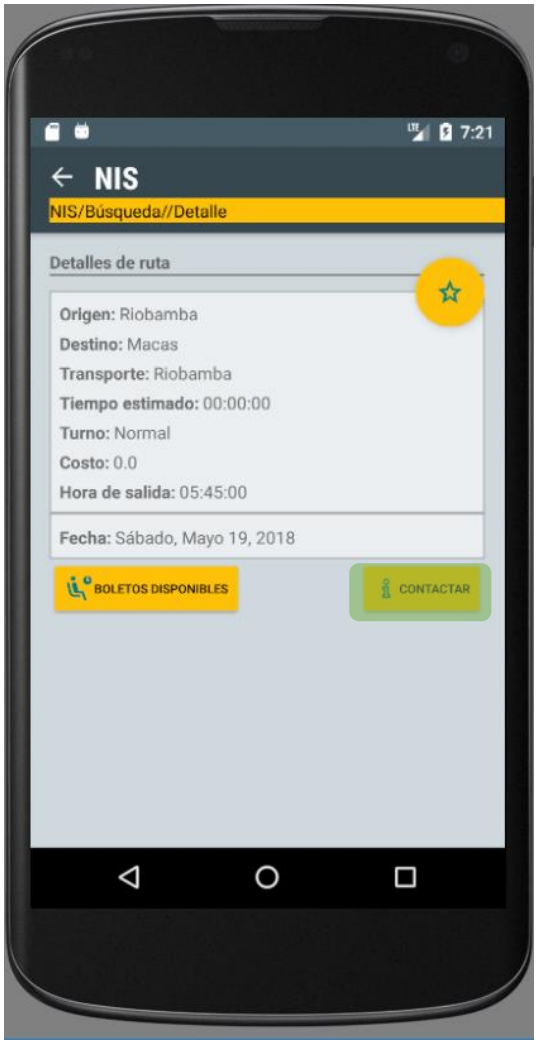
8. Detalles de un horarios específico dentro de una ruta.

Para ver los detalles de un horario específico dentro de una ruta se debe seleccionar uno de la lista, lo cual lo llevará a la pantalla de detalles.

Selección de un horario dentro de una ruta específica.

Realizado por: Wilmer B. 2018.

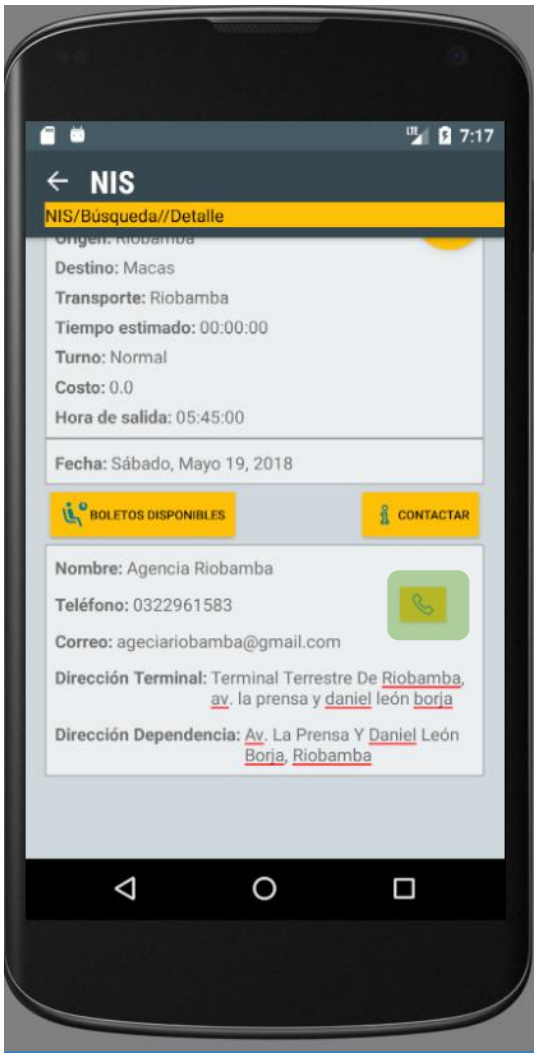
Dentro de esta pantalla se mostrará los detalles del horario seleccionado dentro de una ruta, información como: la ciudad origen, destino, la cooperativa de transporte, el tiempo estimado de viaje, turno, costo, la hora de salida y la fecha actual. Pero esta pantalla también presenta 3 botones, el primero BOLETOS DISPONIBLES, este botón muestra la lista de boletos disponibles en este horario. Botón CONTACTAR, este botón muestra información acerca de la agencia de cooperativa de transporte. Y un tercer botón con el ícono de una estrella el cual representa a favoritos, dando la opción de agregar dicha ruta a favoritos.



Detalle de ruta – contactar.

Realizado por: Wilmer B. 2018.

Al presionar el botón CONTACTAR, muestra la información de la agencia de cooperativa de transporte, información como: nombre de la agencia, teléfono, correo, dirección del terminal en caso de pertenecer a uno y la dirección de la agencia. Además de poder realizar una llamada telefónica a la agencia de cooperativa desde la aplicación, esta llamada consumirá saldo con la operadora.



Llamada telefónica a agencia.

Realizado por: Wilmer B. 2018.

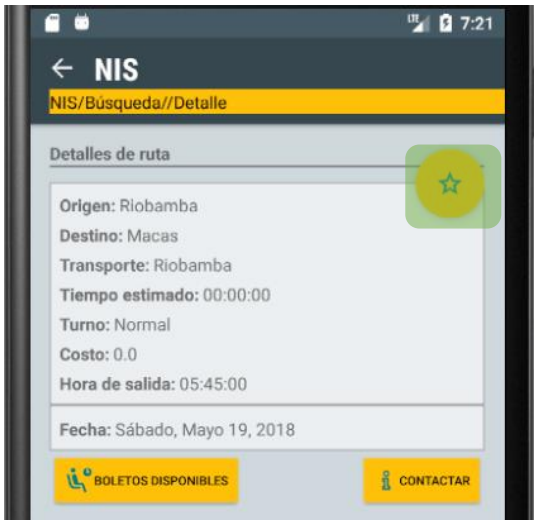
Mientras que, si se da clic en el botón BOLETOS DISPONIBLES, muestra un apartado dentro de la pantalla con la lista de números de boleto disponibles en ese horario, así como también el número de bus en el que se hará dicho recorrido de dicha ruta.

Consulta de boletos disponibles.

Realizado por: Wilmer B. 2018.

9. Agregar un horario des una ruta específica como favorita

Para poder agregar una ruta con un horario específico a favoritos debemos buscar horarios de rutas, luego de elegir un horario debemos presionar sobre este y nos llevará a la pantalla de detalles, en la cual se encuentr ubicado el botón de favorito.



Agregar ruta a favoritos.

Realizado por: Wilmer B. 2018.

Entonces si se desea agregar la presente ruta como favorita, debemos presionar el botón de favoritos y nos presentará un mensaje diciendo que nuestra ruta fue agregada a favoritos.

Mensaje de confirmación de ruta agregada a favoritos.

Realizado por: Wilmer B. 2018.

Para poder verificar o visualizar nuestra ruta favorita debemos ir a la pantalla de inicio y presionar el botón FAVORITOS.

Pantalla de inicio - Listar rutas favoritas.

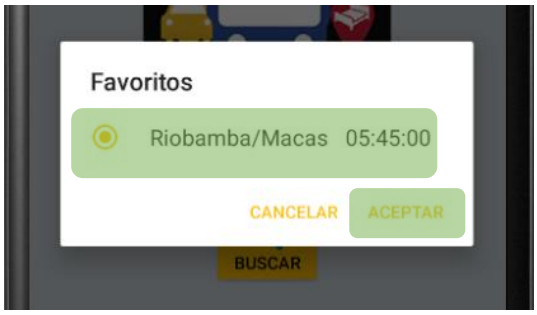
Realizado por: Wilmer B. 2018.

Entonces nos mostrará la lista de rutas favoritas, de la cual podemos seleccionar la que se desee y a continuación ver sus detalles.

Lista de rutas favoritas ordenada.

Realizado por: Wilmer B. 2018.

Cuando se seleccione una de las rutas favoritas y se presione el botón ACEPTAR, nos llevará a la pantalla de detalles de dicha ruta.



Selección de ruta favorita para mostrar detalle.

Realizado por: Wilmer B. 2018.

En la presente pantalla se mostrará la lista de detalles correspondientes a una ruta favorita.

Detalle de ruta favorita.

Realizado por: Wilmer B. 2018.